ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Estudio y análisis de redes neuronales basadas en polinomios ortogonales generados en puntos arbitrarios

AUTORA: Aurora Suárez Vélez

TUTOR: Juan Pascual García

Septiembre 2013

Autora	**Aurora Suárez** Vélez
E-Mail del Autor	**aurorasuarez88@hotmail.com**
Director(es)	**Juan Pascual García**
E-Mail del director	**juan.pascual@upct.es**
Titulo de PFM	**Estudio y análisis de redes neuronales basadas en polinomios ortogonales generados en puntos arbitrarios**

Resumen

Las redes neuronales son una de las técnicas más eficientes en las labores de aproximación de funciones y clasificación de datos. Estas tareas poseen un amplio abanico de aplicaciones en la industria de las Telecomunicaciones. Dentro de los distintos tipos de redes neuronales encontramos las redes neuronales basadas en polinomios ortogonales. En estas redes neuronales se emplean polinomios tales como los polinomios de Chebychev, los polinomios de Lagrange y los polinomios de Hermite. Todos estos polinomios son ortogonales únicamente en una parte del espacio de solución o en un conjunto de datos. Por ejemplo, los polinomios de Chebychev los ortogonales en los ceros de los propios polinomios. Por tanto, estos polinomios aunque útiles en las tareas de aproximación poseen grandes restricciones en su uso. En este proyecto se plantea el uso de polinomios ortogonales en cualquier conjunto de datos; este hecho permite generar redes neuronales muy eficientes y que se pueden emplear en una gran variedad de aplicaciones.

Titulación	**Ingeniero Técnico de Telecomunicación, especialidad Telemática**
Departamento	**Departamento de Tecnologías de la Información y Comunicaciones (TIC)**
Fecha de presentación	**Septiembre 2013**

Índice general

1. Introducción.

1.1. Redes neuronales: definición y tipos.

Podemos definir una red neuronal como un procesador ampliamente distribuido de forma paralela, y compuesto por unidades simples de procesado. Esta estructura tiene una propensión natural para guardar el conocimiento y para hacer disponible su uso [1]. El adjetivo neuronal hace referencia a su parecido con un cerebro humano en dos aspectos:

- La red adquiere el conocimiento de su entorno a través de un proceso de aprendizaje.
- El peso de cada una de las conexiones interneuronales, conocidas como pesos sinápticos, es empleado para almacenar el conocimiento adquirido.

El proceso de aprendizaje consiste en un algoritmo denominado algoritmo de entrenamiento. Su función es la de modificar los parámetros libres, tales como los pesos sinápticos o los parámetros que determinan la función matemática insertada en cada neurona. El algoritmo de entrenamiento de una red neuronal posee la característica propia de generar nuevas neuronas hasta que se alcanza un cierto nivel de conocimiento. Por tanto, durante el entrenamiento se añadirán nuevas neuronas mediante la evaluación de sus parámetros libres, examinando la calidad del aprendizaje cada cierto tiempo o cada cierta cantidad de neuronas. Por lo tanto, durante el entrenamiento las redes neuronales pueden cambiar su topología; este aspecto es similar al nacimiento y muerte de las neuronas del cerebro. Pero hay que decir que el cerebro humano es mucho más complejo que cualquier red neuronal, y por tanto los puntos en común son ciertos a un nivel muy simple.

Todos los tipos de redes neuronales deben estar compuestos por módulos independientes llamados neuronas. Por lo tanto, la estructura de todas las redes neuronales va a ser similar. La diferenciación entre los diferentes tipos de redes neuronales está determinada por el tipo de entrenamiento, por la clase de función matemática que define la neurona, o bien por ambos. Así, aquellas redes neuronales que posean una clase distinta de neuronas, pero compartan una misma estructura, pueden compartir un mismo algoritmo de entrenamiento. Asimismo, una misma estructura con el mismo tipo de neuronas puede poseer diferentes tipos de entrenamiento, dando lugar a dos clases de redes neuronales diferentes. Cada tipo de entrenamiento está fundamentado en una teoría científica o matemática diferente. Para que una técnica de adquisición de conocimiento de un entorno o problema sea catalogada como red neuronal, debe poder ser estructurada en módulos independientes y contar con un método de entrenamiento propio y adecuado. Las características más importantes de las redes neuronales son las siguientes:

* **No-linealidad:** generalmente la mayor parte de los problemas que deben resolver las redes neuronales son no lineales. La red neuronal debe presentar a su vez un carácter no lineal. Dicha propiedad se consigue mediante el uso de neuronas no lineales.

* **Aprendizaje supervisado:** consiste en aprender a partir de ejemplos en los que se provee una entrada y su salida correspondiente. Los parámetros de la red neuronal se ajustan para que la red neuronal represente con cierto grado de fidelidad la relación entre los datos de entrada y los datos de salida. En ocasiones solo ciertos parámetros son calculados teniendo en cuenta los valores de los datos de salida.

* **Adaptabilidad:** las redes neuronales poseen la capacidad de cambiar sus parámetros libres, como por ejemplo sus pesos, para adaptarse a los cambios que surjan en su entorno. El cambio en los parámetros estriba generalmente en un re-entrenamiento en el que se emplean los nuevos datos correspondientes al entorno cambiante. La estructura de ciertos tipos de redes neuronales permite con mayor facilidad la operación de éstas en entornos no-estacionarios. Sin embargo, la capacidad de adaptación no solo es provista por la estructura de la red neuronal sino también por el algoritmo de entrenamiento.

*** Estructura paralela:** toda red neuronal posee una arquitectura en forma de módulos conectados en paralelo. De esta forma, los cálculos realizados en cada una de las neuronas son independientes del resto de neuronas. Esta característica provee a las redes neuronales de dos propiedades muy deseables. En primer lugar presentan una alta tolerancia a fallos. Si aparece un fallo en una neurona o en alguna de sus conexiones, la respuesta se degradará levemente. En segundo lugar, gracias no solo a la estructura paralela, sino también a la tolerancia a fallos, las redes neuronales son aptas para su implantación en tecnología de tipo VLSI ("very large-scale integrated technology"). Además, la paralelización de los cálculos en la red neuronal permite que la respuesta sea rápidamente calculada.

* **Uniformidad en el análisis y diseño:** a pesar de la heterogeneidad de las teorías empleadas para generar los diferentes tipos de redes neuronales, todas ellas comparten una estructura similar en forma de neuronas interconectadas. Esta propiedad permite que diferentes clases de algoritmos de entrenamiento sean en ocasiones compartidas por distintas redes neuronales.

1.1.1. Visión histórica sobre las clases de redes neuronales.

El artículo pionero en el campo de la cibernética y las redes neuronales data de 1943. En [2] se describió por primera vez una red neuronal formada por neuronas simples interconectadas y operando de forma sincronizada. Se demostró que una red constituida de esta forma podía resolver en principio cualquier función computable.

Hasta la década de los 50 no aparece el primer intento de realizar una simulación de una red neuronal en un ordenador. Así en [3] se implantó en un

ordenador la teoría de aprendizaje de Hebb [4]. Esta teoría, siguiendo una idea de Ramón y Cajal, postulaba que la efectividad de una sinapsis se refuerza cada vez que alguna de las neuronas a las que está conectada se activa. Ese mismo año se diseñó una red neuronal con sinapsis o pesos modificables, capaz de clasificar conjuntos simples de patrones binarios [5]. Los avances continuaron hasta el final de la década. Es muy destacable la introducción por parte de Rosenblatt de un nuevo método de aprendizaje supervisado basado en una unidad de cálculo a la que denominó perceptrón [6]. En [7] se demostró el teorema de convergencia del perceptrón. Esta red neuronal se convirtió, debido a estos avances, en la más exitosa de la década de los 60, dando lugar a multitud de aplicaciones. Los perceptrones utilizan como función matemática funciones de tipo logístico, estas funciones posee una parta casi lineal y una parte en la que el valor de salida se satura. Esta década fue fructífera para el campo de las redes neuronales, ya que surgieron además otros tipos de redes como Adaline ("Linear Adaptative Element" o elemento linear adaptativo) [8]. Una evolución de la anterior red neuronal se convirtió en la primera red neuronal formada por más de una capa de neuronas. En efecto, Madaline ("Multiple Adaline") estaba formada por varias capas con múltiples elementos adaptativos [9].

Sin embargo, el desarrollo de nuevas redes neuronales y de sus aplicaciones se frenó en la década de los años 70 debido a diversos factores. Por un lado se incumplieron las elevadas expectativas creadas durante la década anterior. Las redes neuronales podían ser aplicadas en multitud de tareas, pero estaban muy lejos de alcanzar las capacidades del cerebro humano. Por otro lado, en [10] se demostró que una red neuronal basada en una única capa de perceptrones poseía límites fundamentales en su capacidad de cálculo. Además, los autores afirmaron que no existían razones para suponer que un Perceptrón multicapa ("Multilayer Perceptron" o MLP según las siglas inglesas) podría superar las limitaciones de un perceptrón de una sola capa. Finalmente, existían pocos ordenadores disponibles para realizar las simulaciones necesarias para avanzar en el desarrollo de nuevas redes neuronales. Por tanto, solo unos pocos investigadores continuaron el estudio de las redes neuronales. En la siguiente década, gracias a las mejoras tecnológicas, los estudios relacionados con redes neuronales tomaron una nueva importancia. Así, a principios de la década se introdujo un nuevo enfoque en el análisis de redes neuronales recurrentes, gracias al uso de una función de energía para definir su funcionamiento y de la aplicación de teorías provenientes de la física [11]. Las redes recurrentes son aquellas en las que aparece una realimentación desde la salida de datos hasta la entrada. A partir del trabajo mencionado anteriormente, las redes recurrentes suelen ser denominadas como redes de Hopfield. En 1985 el empleo de la teoría de optimización "Simulated Annealing" sirvió para el desarrollo de la primera red multicapa que realmente tuvo éxito en la solución de problemas [12]. A esta red se le denominó máquina de Boltzmann y demostró que la hipótesis de Minsky y Papert no era cierta. Un año más tarde se desarrolló uno de los métodos de aprendizaje más empleados, el algoritmo de retro-propagación ("back-propagation algorithm") [13]. Dicho método ha sido usado hasta el momento presente en multitud de aplicaciones sobre todo en perceptrones multicapa. Una alternativa a esta red neuronal surgió en 1988 con las Redes neuronales basadas en funciones de base radial ("Radial Basis Function Neural Networks" o RBFNN según las siglas inglesas) [14]. Estas redes neuronales posee como función neuronal fundamental a funciones de tipo gaussiano determinadas por una

media y una varianza.

A principios de la década de los 90 aparecieron las Máquinas de vectores soporte (”Support Vector Machines“ o SVM según las siglas inglesas) cuyo funcionamiento está basado en la teoría del aprendizaje con conjuntos de tamaño finito [15, 16]. El diseño de una SVM está relacionado con la dimensión de Vapnik-Chervonenkis (VC), que provee una medida de la capacidad de una red neuronal de aprender a partir de un conjunto de ejemplos. Al igual que las RBFNN, las SVM poseen principalmente como función neuronal funciones de tipo gaussiano.

Durante las dos últimas décadas se han producido numerosos avances en la mejora del proceso de entrenamiento y diseño de diferentes tipos de redes neuronales. Así, se han empleado como funciones neuronales un tipo de funciones ampliamente conocido en el terreno de la aproximación de funciones: los polinomios. Por ejemplo, los polinomios de Hermite han sido empleados en el desarrollo de diversas redes neuronales de estructura similar a la de una RBFNN, tal como se muestra en [17, 18, 19]. Los polinomios ortogonales de Legendre también han sido usados como funciones neuronales, ver [20, 21].

En este proyecto fin de carrera, se van a utilizar los polinomios de Chebychev como función neuronal en una red neuronal. La forma en la que se emplean los polinomios de Chebychev en este proyecto permitirá alcanzar un alto grado de eficiencia en la red neuronal, tal como se demostrará en los sucesivos apartados. Al igual que los mencionados polinomios, los polinomios de Chebychev ya han sido empleados en diversos tipos de redes neuronales. El primer trabajo que utilizó este tipo de polinomios en una red neuronal es [22]. Este trabajo es contemporáneo a la aparición del algoritmo de entrenamiento OLS para RBFNNs. Para entrenar las Redes neuronales de Chebychev utilizaba el bien conocido método de retro-propagación de los MLPs. Un enfoque diferente es aplicado en [23]. En este artículo los polinomios de Chebychev aproximan a las funciones neuronales o funciones de activación de otros tipos de redes neuronales. Cada función neuronal se convierte en una suma de polinomios. De este modo, los polinomios de Chebychev se introducen en la red neuronal original sustituyendo las funciones a las que aproximan. El entrenamiento se limita al cálculo de un conjunto de pesos que agrupan los coeficientes de la aproximación de las funciones neuronales y los pesos de la red neuronal original. Gracias a la capacidad de representación de otros tipos de redes neuronales con capacidades de aproximación universal, el modelo propuesto en [23] basado en polinomios de Chebychev tiene la habilidad de la aproximación universal. Este tipo de red neuronal fue posteriormente aplicado en diversos problemas prácticos [24, 25].

En [26] se desarrolla una red neuronal con una estructura similar a la de una RBFNN. Así, la capa oculta de neuronas está compuesta por polinomios de Chebychev mientras que la capa de salida es lineal. En este caso, la propiedad de ortogonalidad expresada (la veremos posteriormente) es empleada para calcular fácilmente los pesos de la capa lineal. Los trabajos anteriores se centran en el cálculo de los pesos que conforman ciertas capas de la red neuronal. Los polinomios de Chebychev son seleccionados como funciones neuronales por su capacidad en la aproximación de funciones. Sin embargo, ninguno de los trabajos propone un método de selección de

neuronas, es decir, de elección de unos polinomios o combinaciones de polinomios en lugar de otros. En este proyecto fin de carrera mostraremos un método en una red neuronal de Chebychev para seleccionar los pesos y las neuronas de forma óptima.

1.2.Tipos de entrenamiento.

El proceso de aprendizaje es un proceso iterativo, en el cual se va refinando la solución hasta alcanzar un nivel de operación suficientemente bueno. La mayoría de los métodos de entrenamiento utilizados en las redes neuronales con conexión hacia delante consisten en la utilización de una función de error que mida el rendimiento actual de la red en función de los pesos sinápticos. El objetivo del método de entrenamiento es encontrar el conjunto de pesos sinápticos y parámetros de las funciones neuronales que minimizan (o maximizan) la función. El método de optimización proporciona una regla de actualización de los pesos y parámetros que en función de los patrones de entrada modifica iterativamente los pesos hasta alcanzar el punto óptimo de la red neuronal.

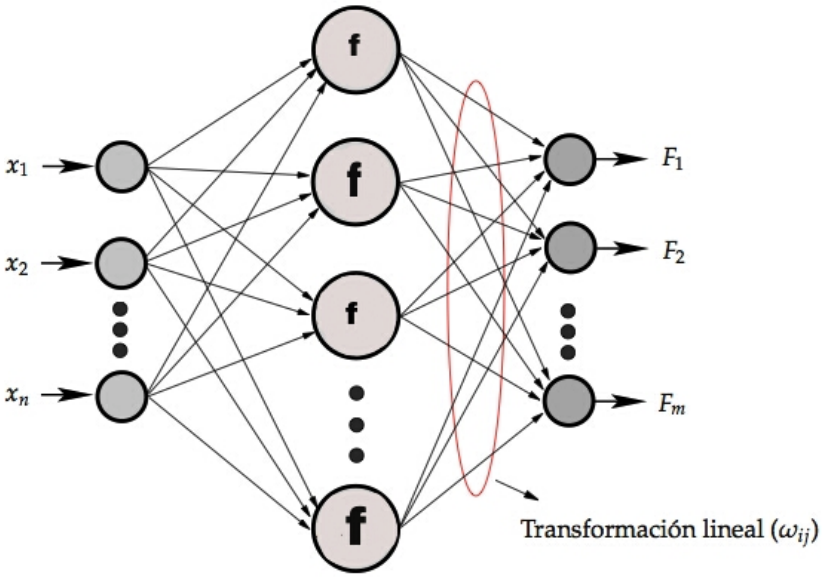


Figura 1.1: Esquema general de una red neuronal.

* **Gradiente:** este método de entrenamiento es el más utilizado. Define una función E(W) que proporciona el error que comete la red en función del conjunto de pesos sinápticos W (ver ecuación 1.1 y figura 1.1). El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error, aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno [27]. El principio general del método es: dado un conjunto de pesos W(0) para el instante de tiempo t=0, se calcula la dirección de máxima variación del error. La dirección de máximo crecimiento de la función E(W) en W(0) viene dado por el gradiente ∇E(W). Luego, se actualizan los pesos siguiendo el sentido contrario al indicado por el gradiente ∇E(W), dirección que indica el sentido de máximo decrecimiento.

De este modo se va produciendo un descenso por la superficie de error hasta alcanzar un mínimo local:

$$W(t+1) = W(t) - \alpha \nabla E(W) \qquad (1.1)$$

donde α indica el tamaño del paso tomado en cada iteración, pudiendo ser diferente para cada peso e idealmente debería ser infinitesimal. El tamaño del paso es un factor importante a la hora de diseñar un método de estas características. Si se toma un paso demasiado pequeño el proceso de entrenamiento resulta muy lento, mientras que si el tamaño del paso es muy grande se producen oscilaciones en torno al punto mínimo.

*** Retro – propagación ("Backpropagation"):** es el método de entrenamiento más utilizado en redes con conexión hacia delante. Es un método de aprendizaje supervisado de gradiente descendente, en el que se distinguen dos fases: primero se aplica un patrón de entrada, el cual se propaga por las distintas capas que componen la red hasta producir la salida de la misma. Esta salida se compara con la salida deseada y se calcula el error cometido por cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capas intermedias. Cada neurona recibe un error que es proporcional a su contribución sobre el error total de la red. Basándose en el error recibido, se ajustan los errores de los pesos sinápticos de cada neurona. Esta técnica minimiza el error promedio al cuadrado entre la salida real y la esperada, aplicando el concepto de gradiente descendiente.

*** OLS:** El algoritmo de Mínimos cuadrados ortogonales ("Orthogonal Least Squares") es una técnica supervisada, es decir, tiene en cuenta la salida de la red neuronal para diseñarla. [28]. método OLS intenta calcular cómo contribuye cada neurona a la salida deseada y al mismo tiempo calcula el peso de cada neurona. Para llevar a cabo esta doble tarea, el método OLS representa la operación realizada por la red neuronal en forma matricial. Así, cada neurona equivale a una columna de una matriz. Esta matriz es multiplicada por un vector o matriz de pesos para formar el vector o matriz de los datos de salida respectivamente. El método OLS consiste en un proceso de ortogonalización de la matriz de neuronas; esta ortogonalización permite seleccionar una a una cada neurona atendiendo a cuál es su contribución a la representación de los datos de salida. En este proyecto se ha elegido este método para diseñar las redes neuronales de Chebychev. Como se explicará en el apartado 2, este método posee una limitación muy importante: si las columnas de la matriz de neuronas no son ortogonales, el método es incapaz de seleccionar el mejor conjunto de neuronas. Únicamente si las columnas son ortogonales se consigue un rendimiento óptimo. Por ello, en este proyecto se han empleado polinomios de Chebychev de primera especie; estos polinomios son ortogonales en ciertos puntos. Esta es sin duda, una limitación de estos polinomios. Por ello, en este proyecto también se han empleado polinomios de

Chebychev que son ortogonales en cualquier conjunto de puntos. De este modo se supera la principal limitación del método OLS en cualquier conjunto de datos.

2. Método de entrenamiento OLS.

2.1. Algoritmo OLS.

La estructura neuronal mostrada en la Figura 1.1 realiza la siguiente operación para calcular la salida si ésta es unidimensional:

$$F = \sum_{i=1}^{M_s} w_i f_i(\mathbf{x}) \tag{2.1}$$

donde **F** es el valor de salida de la red neuronal, **x** es el vector de entrada, M_s es el número de neuronas, f_i es la función no lineal de la neurona i-ésima y w_i es el peso de la neurona i-ésima. En esta estructura neuronal no existen pesos que multipliquen al vector de entrada.

La operación anterior se puede expresar de forma matricial. El conjunto de valores de salida de las funciones neuronales f_i y el conjunto de pesos w_i serán vectores columna. De este modo, la ecuación (2.1) se expresa como:

$$F = \mathbf{f}^T(\mathbf{x}) \cdot \mathbf{w} \tag{2.2}$$

donde el vector traspuesto de **f** es un vector fila. Como ya se comentó en la Introducción para entrenar la red, es decir, para seleccionar las neuronas y calcular los pesos, hay que generar una serie de puntos de entrenamiento. En el algoritmo de entrenamiento OLS es necesario calcular la salida de la red neuronal para cada uno de los puntos de entrenamiento. Esta operación se puede expresar de forma matricial:

$$\mathbf{y} = \mathbf{F} + \mathbf{E} = \mathbf{P}\omega + \mathbf{E} \tag{2.3}$$

donde:

$$\mathbf{y} = [y_1 y_2 \ldots y_N]^T$$

$$\mathbf{P} = [p_1 p_2 \ldots p_{M_S}]^T$$

$$\mathbf{w} = [w_1 w_2 \ldots w_{M_S}]^T$$

$$\mathbf{E} = [\varepsilon_1 \varepsilon_2 \ldots \ \varepsilon_N]^T \tag{2.4}$$

En el conjunto de ecuaciones, la ecuación (2.4) el vector columna **y** es el conjunto de valores deseados, es decir, el conjunto de valores de la función que se desea aproximar, el vector columna **w** es el vector de pesos y el vector **E** son los errores que comete la red neuronal en su aproximación. La matriz **P** está formada por una serie de vectores columna $\mathbf{p}_i$ con $1 \leq i \leq M_S$. Los elementos de cada vector columna son fruto de evaluar la función f, propia de las neuronas, en los puntos del conjunto de entrenamiento. Como se observa en la ecuación (2.4) existen M_S vectores columna obtenidos a partir de un número igual de neuronas (M_S siempre es igual o menor que N, número de puntos de entrenamiento).

A cada vector columna de **P** se le denomina regresor:

$$\mathbf{p}_i = [p_{1i}\ p_{2i}\ \dots\ p_{Ni}]^T\ \ regresor\ \ i-ésimo \tag{2.5}$$

Los M_S regresores forman una base de vectores que generan un espacio. La solución mínimo cuadrática del sistema de la ecuación (2.3) garantiza que la solución encontrada (**Pω**) es la proyección del vector **y** en el espacio generado por los regresores. El cuadrado de la proyección de **y** es parte de la energía de la solución deseada. Debido a que existe un cierto grado de correlación entre los distintos regresores, no es posible decidir cuánto contribuye un regresor individualmente a la energía de la función objetivo. Para seleccionar la mejor combinación de regresores, al existir correlación entre ellos, habría que probar todas las posibles combinaciones, de forma que el entrenamiento así realizado resultaría imposible.

El método OLS transforma el conjunto de regresores p_i en un conjunto o base de vectores ortogonales de forma que se puede estimar la contribución a la energía de salida de cada uno de los vectores ortogonales de forma individual. El algoritmo proporciona así un modo de seleccionar paso a paso solo aquellos regresores (neuronas) que más aporten a la representación de la función objetivo. En consecuencia, es previsible que el número final de neuronas sea reducido, consiguiendo al mismo tiempo un bajo nivel de error en la aproximación de la función objetivo. En [29] se demuestra que el algoritmo OLS no es capaz de proporcionar el conjunto de regresores mínimo para alcanzar un determinado nivel de precisión en una tarea de aproximación. Este hecho será explicado en el siguiente apartado, ahora, en este apartado explicaremos completamente el algoritmo OLS. Este algoritmo se fundamenta en la descomposición de la matriz **P** en el producto de dos matrices [28]:

$$\mathbf{P} = \mathbf{MA} \tag{2.6}$$

donde **A** es una matriz triangular de tamaño $M_S x\ M_S$ con 1's en la diagonal y **M** es una matriz con columnas ortogonales:

$$A = \begin{pmatrix} 1 & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1M_S} \\ 0 & 1 & \alpha_{23} & \cdots & \alpha_{2M_S} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \alpha_{M_S-1M_S} \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

$$M = [m_1\ m_2 \cdots m_{M_S}]\quad m_i = [m_{1i}\ m_{2i} \cdots m_{Ni}]^T \quad 1 \le i \le M_S \tag{2.8}$$

Si sustituimos la ecuación (2.7) en la ecuación (2.8) obtenemos:

$$\mathbf{y} = \mathbf{MAw} + \mathbf{E} \tag{2.9}$$

Si al producto de la matriz **A** por el vector de pesos se le denomina vector **g** podemos expresar la ecuación anterior como:

$$\mathbf{y} = \mathbf{Mg} + \mathbf{E} \tag{2.10}$$

Debido a que la matriz **M** está compuesta por columnas ortogonales la solución mínimocuadrática, $\hat{g}$, de la ecuación (2.10) toma la expresión:

$$\mathbf{M^T y} = \mathbf{M^T M}\hat{\mathbf{g}} = \mathbf{H}\hat{\mathbf{g}} \tag{2.11}$$

Por tanto el vector $\hat{g}$ se despeja como:

$$\hat{\mathbf{g}} = \mathbf{H^{-1}M^T y} \tag{2.12}$$

donde la matriz diagonal **H** está formada por elementos de valor $h_i = m_i^T m_i;\ 1 \le i \le M_S$ en la diagonal y 0's en el resto de posiciones. Su inversa es:

$$H^{-1} = \begin{pmatrix} 1/h_1 & 0 & \cdots & 0 \\ 0 & 1/h_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/h_{M_S} \end{pmatrix} \tag{2.13}$$

Por lo tanto cada elemento del vector $\hat{g}$ se expresa como:

$$\hat{g}_i = \frac{\mathbf{m_i^T y}}{h_i} = \frac{\mathbf{m_i^T y}}{\mathbf{m_i^T m_i}} \tag{2.14}$$

Los pesos se calculan a partir del sistema triangular:

$$\mathbf{A}\hat{\omega} = \hat{\mathbf{g}} \tag{2.15}$$

El sistema anterior y el conjunto de vectores ortogonales de la ecuación (2.8) se puede obtener mediante el empleo del método de Gram-Schmidt. Este procedimiento calcula en cada iteración una columna de la matriz **A** y convierte en ortogonal un vector de **P** respecto al conjunto de vectores ya ortogonalizados. El método OLS aplicado a la selección de un conjunto de regresores hace uso del algoritmo de Gram-Schmidt para incorporar a la base ortogonal aquel candidato que más aporte a la energía de salida. En la iteración *k*-ésima se calcula el resultado de ortogonalizar cada uno de los regresores restantes respecto de los $k - 1$ vectores ya ortogonalizados. Aquel regresor que una vez ortogonalizado contribuya en mayor medida a la energía de la función objetivo, será seleccionado. La contribución a la energía de salida se calcula con el candidato ortogonalizado ya que de este modo se

encuentra incorrelado respecto a los regresores ya seleccionados. Así, su aportación a la función objetivo se evalúa de forma clara.

Para estimar la contribución de cada regresor ortogonalizado a la función o salida deseada comenzamos por definir la energía de la función objetivo como $\mathbf{y^T y}$. Sustituyendo la expresión de la función objetivo de la ecuación (2.9) en la expresión de la energía de salida obtenemos:

$$\mathbf{y^T y} = \hat{\mathbf{g}}\mathbf{M^T M}\hat{\mathbf{g}} + \mathbf{E^T E} \tag{2.16}$$

Como las columnas m_i de la matriz **M** son ortogonales los productos $m_i m_j$ son iguales a cero cuando $i \neq j$. Por lo tanto si se introduce la ecuación (2.14) en la ecuación (2.16), la energía de la función objetivo de salida equivale a:

$$\mathbf{y^T y} = \sum_{i=1}^{M} \hat{\mathbf{g}}^{\mathbf{2}}\, \mathbf{m_i^T m_i} + \mathbf{E^T E} \tag{2.17}$$

El término $\mathbf{E^T E}$ es el error en la aproximación, es decir, la parte de la energía de salida no expresada por los regresores. En cambio, el término $\sum_i^M \hat{\mathbf{g}}_i^2 \mathbf{m}_i^T \mathbf{m}_i$ corresponde a la parte de energía expresada por los regresores ortogonalizados. Así, se puede definir un ratio de reducción del error en la aproximación para cada regresor ortogonalizado:

$$err_i = \frac{\hat{g}_i^2\, \mathbf{m}_i^T \mathbf{m}_i}{\mathbf{y}^T \mathbf{y}} \quad Para\ 1 \le i \le M_S \tag{2.18}$$

Cuanto mayor sea err_i mayor será la contribución del regresor ortogonalizado a la expresión de la energía de la función objetivo. El espacio generado por los regresores ortogonalizados es el mismo que el espacio de los regresores originales; por tanto el proceso sirve para seleccionar los centros de las neuronas. El proceso de entrenamiento comienza con la selección del primer regresor respecto al que van a ser ortogonalizados todos los demás regresores seleccionados. Para simplificar la notación consideramos que $g = \hat{g}$

$$\begin{aligned}
&En\ la\ primera\ iteración\ k = 1 \\
&Para\ 1 \le i \le M_S\ se\ calcula \\
&\mathbf{m_i^1} = \mathbf{p_i}\ vector\ ortogonal\ provisional \\
&g_i^1 = \frac{(\mathbf{m_i^1})^\mathbf{T}\mathbf{y}}{(\mathbf{m_i^1})^\mathbf{T}\mathbf{m_i^1}} \\
&err_i^1 = \frac{(g_i^1)^2 (\mathbf{m_i^1})^\mathbf{T}\mathbf{m_i^1}}{\mathbf{y^T y}}
\end{aligned} \tag{2.19}$$

El superíndice en las ecuaciones anteriores indica la iteración mientras que el subíndice indica el regresor (columna de la matriz **M**) para el que se calcula el ratio de error. Una vez calculados los ratios de error de la primera iteración err_i^1 se busca el regresor que produce la máxima reducción:

$$err_{i_1}^1 = máx\{err_i^1,\ \ 1 \le i \le M_S\} \tag{2.20}$$

Así se selecciona:

$$m_1 = m_{i_1}^1 = p_{i_1} \tag{2.21}$$

En la iteración *k*-ésima (*k* ≥ 2) se habrán ortogonalizado *k* − 1 regresores y se tendrá que seleccionar el *k*-ésimo regresor para que sea incluido en la base de vectores ortogonales. El conjunto de regresores candidatos está compuesto por los regresores no ortogonalizados hasta la presente iteración, es decir, por $M_S - (k-1)$ regresores. El regresor candidato seleccionado será aquel que, una vez ortogonalizado, produzca la mayor reducción del error; el parámetro de la ecuación 2.18 siempre se comprueba con el regresor una vez ortogonalizado ya que así se encuentra incorrelado con los regresores ortogonalizados. Por ello, la contribución del candidato ortogonalizado será independiente de la contribución de los regresores ya seleccionados y ortogonalizados.

A partir de la iteración *k* ≥ 2 se deben calcular los valores α_{jk} de la matriz **A**, ver ecuación (2.7) ya que son necesarios para ortogonalizar el regresor candidato respecto a los (*k* − 1) regresores previamente ortogonalizados. El subíndice *k* indica tanto la columna como la iteración. Dichos valores α_{jk} se calculan gracias a la expresión de la ecuación 2.6

$$\mathbf{M^T P = M^T M A = H A} \qquad \mathbf{A = H^{-1} M^T P} \tag{2.22}$$

A partir de la ecuación anterior, los coeficientes α_{jk}^i para cada uno de los regresores candidatos se calculan como:

$$\alpha_{jk}^i = \frac{\mathbf{m_j p_i}}{\mathbf{m_j^T m_j}} \qquad 1 \le j \le k-1 \tag{2.23}$$

En la ecuación 2.23 el regresor candidato p_i no tiene superíndice ya que dicho regresor no depende de la iteración, su valor es el mismo para cualquier iteración. El superíndice *i* recorre los regresores desde $1 \le i \le M_S$ exceptuando aquellos regresores previamente seleccionados $i \ne i_1 \ne i_2 \ne \llcorner \ \ne i_{k-1}$.

Una vez obtenidos los valores α_{jk}^{i} propios del regresor bajo estudio se calcula la reducción del error de la siguiente forma [28]:

$$\mathbf{m}_i^k = \mathbf{p}_i - \sum_{j=1}^{k-1} \alpha_{jk}^{i} \mathbf{m}_j \qquad \text{ortogonalización del candidato}$$

$$g_i^k = \frac{(\mathbf{m}_i^k)^T \mathbf{y}}{(\mathbf{m}_i^k)^T \mathbf{m}_i^k} \tag{2.24}$$

$$err_i^k = \frac{(g_i^k)^2 (\mathbf{m}_i^k)^T \mathbf{m}_i^k}{\mathbf{y}^T \mathbf{y}} = \frac{[(\mathbf{m}_\mathbf{i}^\mathbf{k})^\mathbf{T} \mathbf{y}) / ((\mathbf{m}_\mathbf{i}^\mathbf{k})^\mathbf{T} \mathbf{m}_\mathbf{i}^\mathbf{k})]^2 (\mathbf{m}_\mathbf{i}^\mathbf{k})^\mathbf{T} \mathbf{m}_\mathbf{i}^\mathbf{k}}{\mathbf{y}^\mathbf{T} \mathbf{y}} = \frac{((\mathbf{m}_\mathbf{i}^\mathbf{k})^\mathbf{T} \mathbf{y})^2}{((\mathbf{m}_\mathbf{i}^\mathbf{k})^\mathbf{T} \mathbf{m}_\mathbf{i}^\mathbf{k}) \mathbf{y}^\mathbf{T} \mathbf{y}}$$

Al igual que en la primera iteración, se busca el regresor que produce la máxima reducción entre los regresores candidatos:

$$err_{i_k}^k = máx\{err_i^k, \;\; 1 \le i \le M_S, \;\; i \ne i_1 \ne i_2 \ne \cdots \ne i_{k-1}\} \tag{2.25}$$

Hallado el valor máximo de error de la iteración *k*-ésima el nuevo vector ortogonal es:

$$\mathbf{m}_k = \mathbf{m}_{i_k}^k = \mathbf{p}_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} \mathbf{m}_j \;\; con \;\; \alpha_{jk} = \alpha_{jk}^{i_k} \;\; y \;\; 1 \le j < k \tag{2.26}$$

El algoritmo finaliza cuando se seleccionan *M* regresores del subconjunto inicial M_S de modo que 1 ≤ *k* ≤ *M*. Como se ha visto, el anterior algoritmo está diseñado para seleccionar un conjunto de regresores cuando la función de salida es unidimensional. Además, el algoritmo halla los pesos de la capa de salida de la red neuronal. El algoritmo se ha explicado para red cuyo espacio o capa de salida unidimensional; esto es debido a que en el presente proyecto todas las funciones utilizadas poseen un espacio de salida unidimensional. El algoritmo se puede utilizar con espacios de salida multidimensionales de manera sencilla.

2.2. Eficiencia del algoritmo OLS.

Como hemos visto en el apartado anterior el algoritmo de entrenamiento OLS selecciona de forma supervisada las neuronas de la red. Cada neurona posee una función no lineal que produce un valor de salida cuando la neurona es alimentada con un vector o valor de entrada. En este proyecto fin de carrera se han utilizado como funciones no lineales polinomios de Chebychev.

Durante el entrenamiento cada neurona, es decir, cada polinomio o combinación de polinomios produce un vector columna que denominamos regresor. El algoritmo selecciona en cada iteración aquel regresor que aporta el mayor incremento en la representación de la energía de la salida deseada. Desde otro punto de vista, se selecciona el regresor que maximiza la reducción del error en la representación de la energía de la salida deseada. Para calcular la contribución de cada regresor a la energía de salida el algoritmo OLS lleva a cabo un proceso de ortogonalización sobre la matriz

original de regresores **P**. En cada paso del proceso de ortogonalización se añade a la base ortogonal aquel candidato que maximiza el parámetro: $\hat{g} = \frac{\mathbf{m_i}^T \mathbf{y}}{\mathbf{m_i}^T \mathbf{m_i}}$. Este valor indica el incremento en la expresión de la energía de la salida.

En la anterior ecuación el vector $\mathbf{m}_i$ es el resultado de ortogonalizar el regresor candidato $\mathbf{p}_i$ respecto a los vectores que forman la base ortogonal. El algoritmo OLS parece proporcionar un mecanismo para medir la contribución de cada regresor $\mathbf{p}_i$ a la energía de salida de forma independiente gracias al proceso de ortogonalización. Podría parecer que esta característica permitiría hallar la red neuronal con el número mínimo de neuronas para alcanzar un determinado nivel de error. Pero este hecho no es cierto. En efecto, en [29] se demuestra que el algoritmo OLS no es capaz generar la red neuronal más compacta en cualquier tarea de aproximación. Este hecho es debido a que originalmente los regresores $\mathbf{p}_i$ no son ortogonales y por tanto las contribuciones de dichos regresores a la energía de salida se encuentran mezcladas.

De este modo, aunque el método OLS calcula el regresor que permite el máximo alineamiento con la salida deseada en una iteración determinada, no es capaz de proveer la mejor representación de la salida en sentido global. Una serie de regresores seleccionados mediante este procedimiento no formará la red neurona más pequeña ya que es muy probable que exista un conjunto de regresores que, de forma conjunta, representen de manera más precisa la salida objetivo.

El problema del algoritmo OLS radica en que selecciona el regresor de forma individual, sin atender a la contribución que realiza de forma global con otros regresores. A pesar de ser un algoritmo de entrenamiento basado en un proceso de ortogonalización, las neuronas formarán regresores que no son ortogonales. El proceso de ortogonalización sirve únicamente para calcular la aportación individual de cada regresor de forma independiente. Dicha independencia puede producir un exceso de neuronas ya que no se tienen en cuenta las propiedades globales de los distintos grupos de regresores que se pueden elegir para componer la red neuronal. En conclusión, el máximo alineamiento con la salida en cada una de las iteraciones no implica que el alineamiento global producido por la red neuronal sea máximo.

El siguiente ejemplo, extraído de [29], muestra la incapacidad del algoritmo OLS para seleccionar el número mínimo de neuronas para un determinado nivel de error.

El problema consiste en aproximar de forma eficiente el vector salida **y** con la matriz de regresores **P** siguientes:

$$y = \begin{pmatrix} -100 \\ 100 \\ 100 \end{pmatrix} \tag{2.27}$$

$$P = \begin{pmatrix} 1 & 0{,}606531 & 0{,}135335 \\ 0{,}606531 & 1 & 0{,}606531 \\ 0{,}135335 & 0{,}606531 & 1 \end{pmatrix} \tag{2.28}$$

La energía de la salida es igual a $\mathbf{y}^{\mathrm{T}}\mathbf{y} = 30000$. El objetivo es conseguir una energía de salida de la red neuronal lo más cercana al valor anterior con el menor número de neuronas, es decir, de regresores. El método OLS selecciona en primer lugar el tercer regresor de la matriz **P** debido a que produce el máximo alineamiento con la salida. En la siguiente iteración se ortogonalizan las columnas primera y segunda de esta matriz, y se calcula el alineamiento de cada vector ortogonal con la salida. El primer regresor ortogonal proporciona el máximo alineamiento, de modo que es seleccionado en segundo lugar. Así, la matriz **P** de regresores ortogonales (también normalizados) es:

$$M_{OLS} = \begin{pmatrix} 0{,}1149 & 0{,}8973 & -0{,}4263 \\ 0{,}5152 & 0{,}3130 & 0{,}7979 \\ 0{,}8494 & -0{,}3113 & -0{,}4263 \end{pmatrix} \tag{2.29}$$

La contribución a la energía de salida de cada uno de los regresores es igual a $\hat{g}_1^2 = 15614{,}1$ $\hat{g}_2^2 = 8019{,}8$ y $\hat{g}_3^2 = 6366{,}2$. El algoritmo OLS efectúa una permutación sobre el orden de los regresores originales de modo que la nueva matriz P obtenida es:

$$P_{OLS} = \begin{pmatrix} 0{,}135335 & 1 & 0{,}606531 \\ 0{,}606531 & 0{,}606531 & 1 \\ 1 & 0{,}135335 & 0{,}606531 \end{pmatrix} \tag{2.30}$$

Si deseamos generar una red neuronal con una única neurona deberemos seleccionar la primera columna de la matriz de la ecuación (2.30).

En cambio, si deseamos emplear dos regresores (neuronas), eliminaríamos la tercera columna de la anterior matriz. En este caso, el porcentaje de la energía representada o expresada por los dos regresores elegidos es cercana al 80 %:

$$\frac{\hat{g}_1^2 + \hat{g}_2^2}{y^T y} \approx 0,79 \tag{2.31}$$

Sin embargo, los dos primeros regresores de la matriz original **P** permiten una representación mucho más precisa de la salida. En efecto, si ortogonalizamos la matriz original sin alterar la posición de los regresores obtenemos, es decir, si ortogonalizamos los regresores en orden (primero el primer regresor como si hubiera sido el regresor seleccionado):

$$M_{opt} = \begin{pmatrix} 0,8494 & -0,4521 & 0,2724 \\ 0,5152 & 0,5976 & -0,6144 \\ 0,1149 & 0,6622 & 0,7405 \end{pmatrix} \tag{2.32}$$

Los regresores de esta matriz ortogonal poseen una contribución a la energía de salida de $\hat{g}_{opt1}^2 = 480,7$ $\hat{g}_{opt2}^2 = 29305,3$ y $\hat{g}_{opt3}^2 = 213,9$. El porcentaje de contribución de los dos primeros regresores alcanza el 99% de la energía de salida:

$$\frac{\hat{g}_1^2 + \hat{g}_2^2}{y^T y} \approx 0,99 \tag{2.33}$$

Esta mejor precisión es debida a que el primer regresor es casi ortogonal respecto al vector de salida. Así, el segundo regresor, al ser ortogonalizado respecto al primero, produce un vector casi paralelo a la salida. El algoritmo OLS es incapaz de detectar esta situación ya que busca en cada iteración el mejor regresor. Por tanto, no selecciona en la iteración inicial el primer regresor sino el tercero, perdiendo por ello la oportunidad de representar fielmente la salida con tan solo dos regresores. La eficiencia del método depende del nivel de error que se desee conseguir en la aproximación. Si se desea representar como mínimo el 50 % de la energía de la salida el algoritmo OLS requiere únicamente el primer regresor seleccionado p_3. En cambio, la matriz original necesita los dos primeros regresores ya que si se emplea únicamente el primero se obtiene un porcentaje de expresión de la energía de salida de tan solo el 1,6 %. No obstante, si el conjunto de entrenamiento no es muy reducido y el nivel de error no es muy elevado, el algoritmo OLS no suministrará el número mínimo de neuronas.

Aunque en este ejemplo la función $\hat{g}_i^2$ disminuye en cada iteración, en general, la aportación a la energía de salida no es una función monótona. Una solución aparente consistiría en ordenar los regresores ortogonales generados por el algoritmo OLS y rechazar aquellos que ofrezcan los menores valores de $\hat{g}_i^2$. No obstante, esta

permutación sobre los regresores ortogonales impide recuperar los regresores originales. Cualquier permutación en las columnas de la matriz **P** conduce a la obtención de una matriz **M** diferente. De este modo, las energías de los regresores solo tienen sentido para un orden particular y concreto de los regresores originales. Para que los regresores posean una función de representación de la energía objetivo $\hat{g}_i^2$ que pueda ser ordenada de forma monótona decreciente es necesario que sean inicialmente ortogonales. Cuando esto ocurre, la proyección de cada vector sobre el vector de salida es independiente del resto de proyecciones, es decir, de contribuciones. De esta forma, la contribución energética de cada uno de los regresores se halla desacoplada del resto de aportaciones. En este caso, el conjunto total de regresores puede ser ordenado según la aportación de cada regresor a la expresión de la función de salida.

Por consiguiente, en una matriz **P** de regresores ortogonales, para un cierto nivel de error de aproximación, el menor conjunto de regresores siempre es aquel compuesto por los *M* primeros regresores del conjunto ordenado que son capaces de superar el error fijado. Ninguna otra selección de los regresores de la matriz original podrá mejorar la precisión del anterior subconjunto sin emplear un número mayor de regresores. Asimismo, un subconjunto de regresores compuesto por *M* regresores siempre tendrá un error de aproximación superior al logrado por los *M* primeros regresores del conjunto ordenado por energías $\hat{g}_i^2$ decrecientes de forma monótona. Por esta razón, en el presente proyecto se han utilizado polinomios de Chebychev de primera especie y polinomios de Chebychev generales. Los primeros son ortogonales en un conjunto de puntos particular, mientras que los segundos son ortogonales en cualquier conjunto de puntos con las limitaciones que se mostrarán en este proyecto. De esta forma, gracias a los polinomios de Chebychev el algoritmo OLS alcanza un rendimiento óptimo.

3. Redes neuronales de Chebychev unidimensionales.

3.1. Polinomios de Chebychev.

Los polinomios de Chebychev son un grupo de polinomios que presentan ortogonalidad continua y discreta en el intervalo [−1, 1]. En particular, los polinomios de Chebychev de primera especie poseen las siguientes expresiones según el orden del polinomio, [30]:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$

…

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \qquad n \geq 1 \tag{3.1}$$

La última línea de la ecuación (3.1) es la fórmula recursiva que puede utilizarse para calcular cualquier polinomio de Chebychev de primera especie (en adelante los denominaremos polinomios de Chebychev). Una de las principales virtudes de este tipo de polinomios radica en que se pueden calcular sin recurrir a ecuaciones recursivas gracias a la siguiente fórmula trigonométrica:

$$T_n(x) = \cos(n \ arc \ \cos(x)) \tag{3.2}$$

En la figura (3.1) se observan los cinco primeros polinomios de Chebychev. Se aprecia que todos los polinomios están limitados entre ±1 de modo que los máximos son iguales a 1 y los mínimos a −1.
El polinomio de Chebychev de orden *N* posee *N* + 1 extremos, máximos y mínimos, en el intervalo (−1, 1) localizados en:

$$x_m = \cos\left(\frac{\pi m}{n}\right) \qquad m = 0,1,\ldots,n \tag{3.3}$$

Las raíces de los polinomios de Chebychev también son calculadas de forma analítica. Así el polinomio de $T_n(x)$ tiene *N* ceros localizados en:

$$x_k = \cos\left(\frac{\pi(k - 1/2)}{N}\right) \qquad k = 1,\ldots,N \tag{3.4}$$

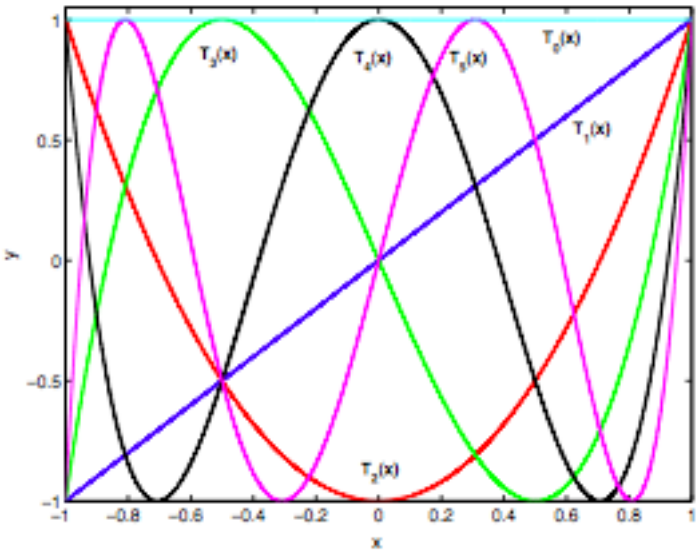


Figura 3.1: Los cinco primeros polinomios de Chebychev en el rango [-1,1].

En estos puntos se satisface la ortogonalidad discreta entre los polinomios de Chebychev. En efecto, los *N* primeros polinomios de Chebychev (desde $T_0(x)$ hasta $T_{N-1}(x)$) son ortogonales en los *N* ceros (x_k) del polinomio $T_N(x)$, [30]. Al polinomio $T_N(x)$ lo llamamos polinomio de orden superior respecto a los polinomios que tienen un orden inferior (orden de 0 a N-1).

$$\sum_{k=1}^{N} T_i(x_k)T_j(x_k) = \begin{cases} 0 & i \neq j \\ m/2 & i = j \neq 0 \\ m & i = j = 0 \end{cases} \tag{3.5}$$

La ortogonalidad discreta de los polinomios de Chebychev es utilizada para generar la bien conocida fórmula de aproximación de Chebychev [30, 31]. Una función *f* (*x*) es aproximada por la siguiente suma de polinomios:

$$F(x) \cong \left[\sum_{j=1}^{N} c_j T_{j-1}(x) \right] - \frac{1}{2} \tag{3.6}$$

donde $x \in [-1, 1]$ y los coeficientes c_j se definen como:

$$c_j = \frac{2}{N}\sum_{k=1}^{N} f(x_k)T_{j-1}(x) = \\ = \frac{2}{N}\sum_{k=1}^{N} F\left[\cos\left(\frac{\pi(k-1/2)}{N}\right)\right]\cos\left(\frac{\pi(j-1)(k-1/2)}{N}\right) \tag{3.7}$$

En la expresión (3.7) se utiliza la representación trigonométrica de los polinomios de Chebychev mostrada en (3.2). Los puntos x_k corresponden a las raíces del polinomio $T_n(x)$ de la ecuación (3.4). De este modo, los *N* primeros polinomios de la fórmula (3.6) son ortogonales, permitiendo una sencilla evaluación de los coeficientes c_j. Mediante un cambio de variable se puede aproximar una función que tenga límites arbitrarios $\left[x_{\inf}, x_{\sup}\right]$, ver [30]:

$$x = \frac{x_{or} - \frac{1}{2}(x_{\sup} + x_{\inf})}{\frac{1}{2}(x_{\sup} - x_{\inf})} \tag{3.8}$$

donde x_{or} representa la variable independiente de la función objetivo definida en el rango original y *x* es el valor correspondiente en el rango de trabajo [−1, 1].

Como se comentó en el capítulo anterior, si las funciones neuronales, es decir, los polinomios son ortogonales, el algoritmo OLS alcanzará su máxima eficiencia. Por lo tanto, si los puntos de entrenamiento de la red neuronal que utiliza polinomios de Chebychev son los ceros del polinomio de Chebychev de orden superior, el algoritmo OLS tendrá un comportamiento óptimo.

3.2 Estudio de las redes neuronales de Chebychev.

En este apartado vamos a estudiar el rendimiento de las redes neuronales de Chebychev (polinomios de primera especie) en tres funciones en las que el espacio de entrada y salida es unidimensional. Estas tres funciones han sido utilizadas en diversos artículos [26] para poner a prueba la capacidad de aproximación de diversas redes neuronales. Como veremos a continuación, estas funciones poseen fuertes variaciones de modo que la capacidad de la red neuronal será puesta a prueba de forma exigente.

Las ecuaciones de las funciones que vamos a estudiar son las siguientes:

$$y(x) = \sin(x) \cdot \cos(2x) \tag{3.9}$$

$$y(x) = 0.2\left[1 + \frac{1}{10}(x-7)^2\right]\cos(2x) + 0.5e^{-2x}\sin(2x - 0.1\pi) \tag{3.10}$$

$$y(x) = \sin(2\pi \cdot 5x) + \sin(2\pi \cdot 10x) \tag{3.11}$$

A continuación vamos a mostrar una gráfica de cada función en el intervalo estudiado; éstas gráficas, sólo tenemos que tener en cuanta la función (color azul), ya que la aproximación con los mejores regresores no es necesaria.

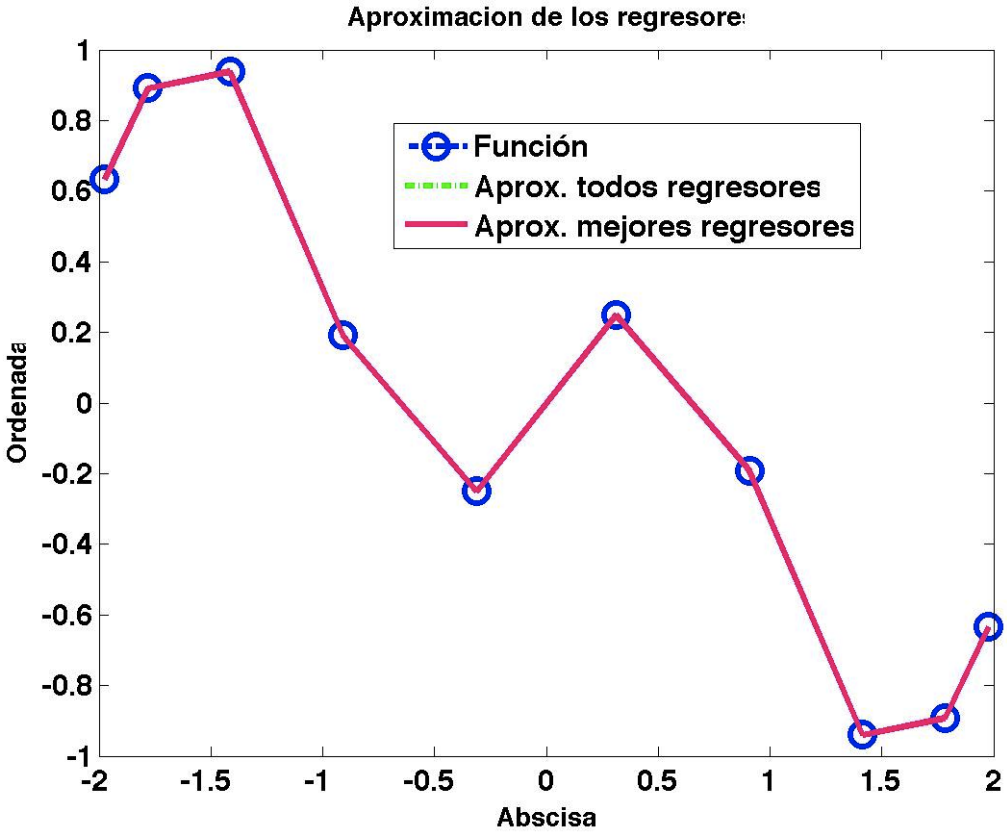


Figura 3.2. Función 3.9 estudiada en el intervalo [-2,2].

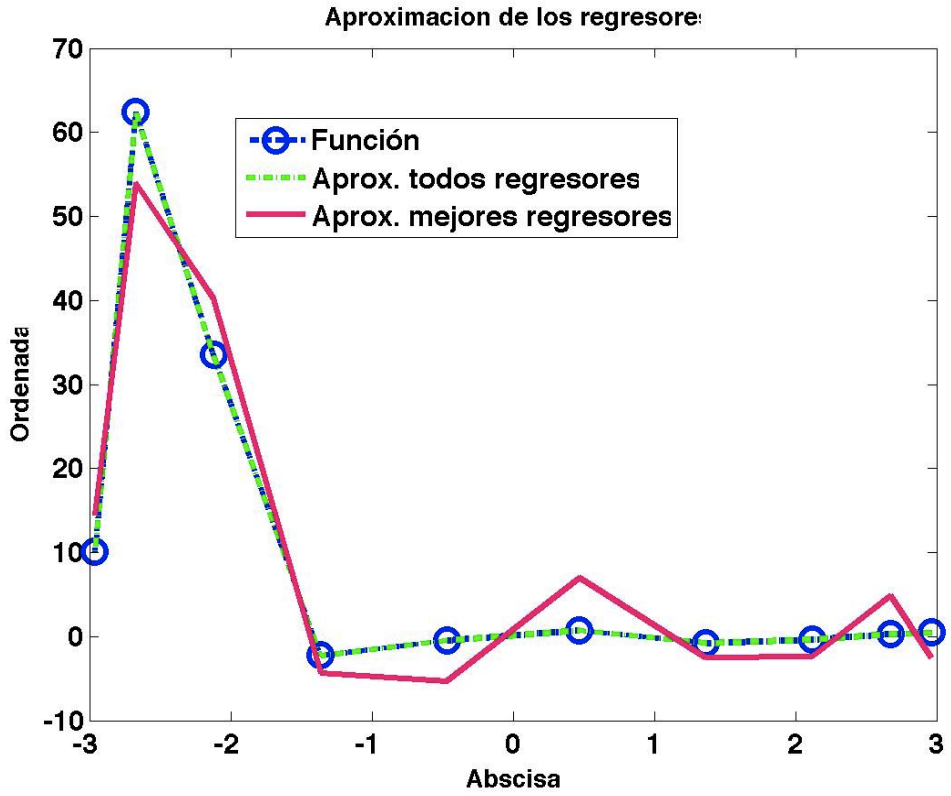


Figura 3.3. Función 3.10 estudiada en el intervalo [-3,3].

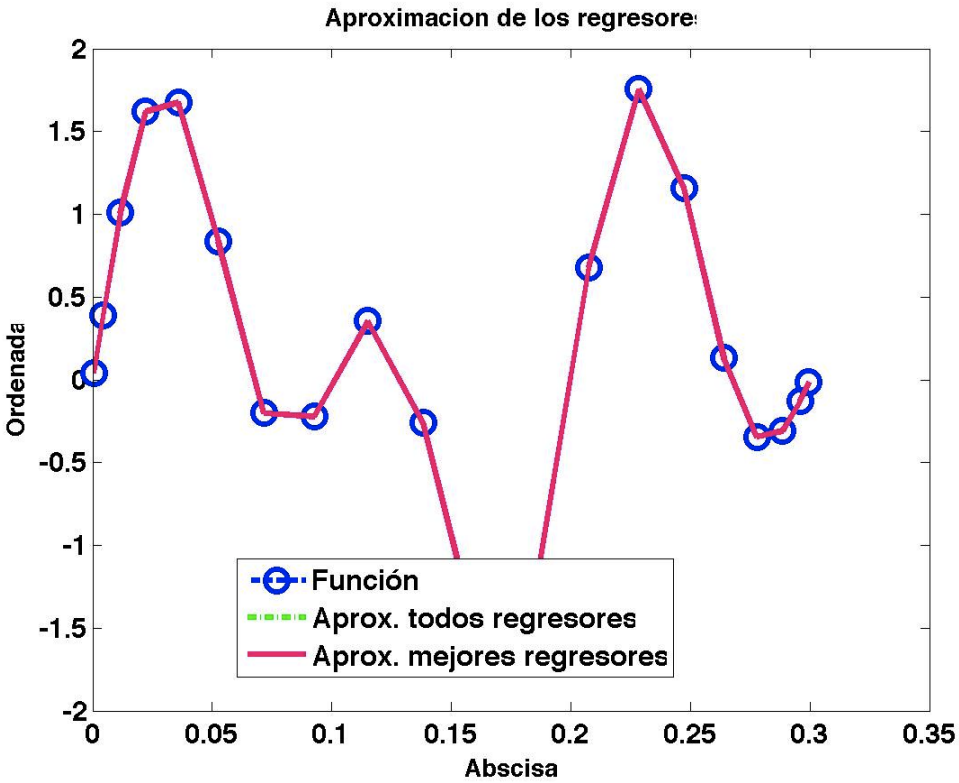


Figura3.4. Función 3.11 estudiada en el intervalo [0,0.3].

3.2.1. Estudio de la ortogonalidad de los regresores (matriz H).

En este apartado vamos a estudiar que efectivamente los polinomios de Chebychev son ortogonales en los intervalos estudiados en los puntos de entrenamiento seleccionados. Cada polinomio de Chebychev forma una neurona de la red neuronal, por lo tanto, cada regresor debe ser ortogonal al resto de regresores. Así, la matriz de regresores **P** estará formada por columnas ortogonales y la matriz **H** (ver ecuación 2.11 y 2.13 del capítulo anterior) será una matriz diagonal.

Para ello vamos a mostrar en gráficas la matriz **H** de las pruebas realizadas, también mostraremos un parámetro que nos indicará el grado en el que la matriz **H** es ortogonal. Este parámetro es el resultado de hacer el cociente entre la media de los elementos de la diagonal y la media de los elementos que no están en la diagonal.

- Tabla y gráficas del estudio de la función 3.9

En la tabla 3.1 se muestran los datos para medir la ortogonalidad de los regresores en la función 3.9. El número de puntos de entrenamiento n, varía entre 5 y 25, y el intervalo (rango de trabajo) de esta función es [-2,2].

PRUEBA	Inicial	Final	n
1	-2	2	5
2	-2	2	10
3	-2	2	15
4	-2	2	20
5	-2	2	25

Tabla 3.1. Datos de las pruebas para medir la ortogonalidad de los regresores en la función 3.9.

En la siguiente tabla mostramos los resultados que miden si la matriz H es diagonal en las pruebas realizadas.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	3.0000	3.775e-16	7.948e+15
2	5.5000	2.808e-15	1.959e+15
3	8.0000	5.999e-15	1.334e+15
4	10.5000	2.903e-15	3.617e+15
5	13	7.212e-15	1.803e+15

Tabla 3.2. Parámetro de medición de la ortogonalidad (ratio diagonal) en la función 3.9.

Gráficas:

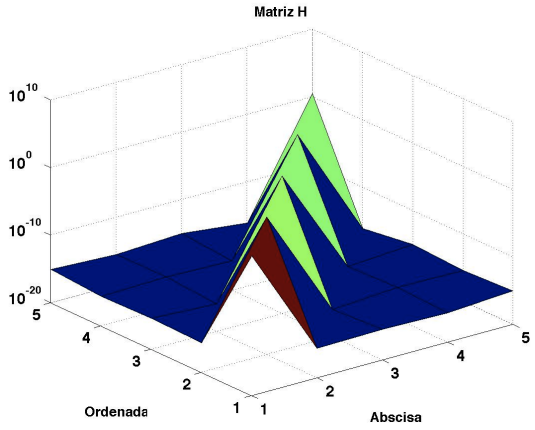


Figura 3.5. Con 5 puntos de entrenamiento (prueba 1).

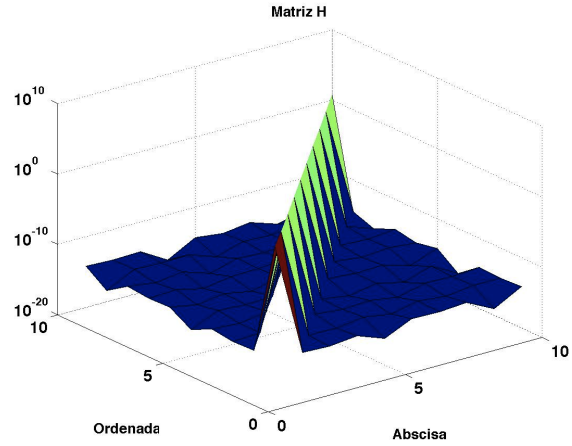


Figura 3.6. Con 10 puntos de entrenamiento (prueba 2).

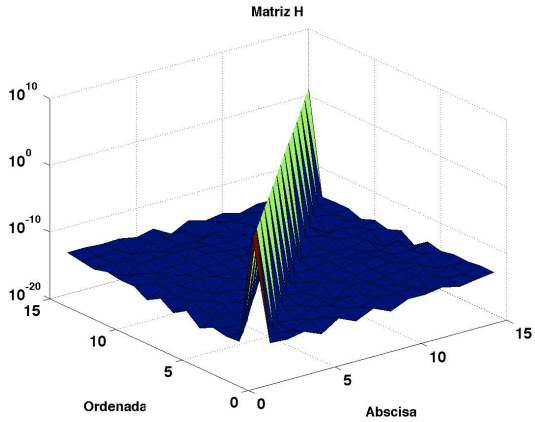


Figura 3.7. Con 15 puntos de entrenamiento (prueba 3).

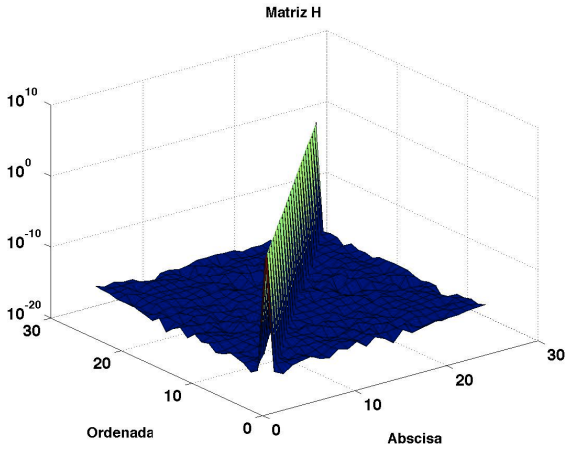


Figura 3.8. Con 25 puntos de entrenamiento (prueba 4).

Tal como se observa en las figuras anteriores (3.5 a la 3.8) y en la tabla 3.2, vemos que la matriz **H** es diagonal y que por tanto los regresores son ortogonales. El valor del parámetro que hemos definido como ratio diagonal cuanto mayor sea, indica que la matriz **H** es mas diagonal. En la tabla 3.2 queda claro que el valor de ratio diagonal es elevado en todos los casos debido a que la media de la diagonal es distinta de cero y la media de los elementos fuera de la diagonal es prácticamente cero.

- Tabla y gráficas del estudio de la función 3.10

En la tabla 3.3 se muestran los datos para las pruebas para medir la ortogonalidad de los regresores en la función 3.10. El número de puntos de entrenamiento n, varía entre 5 y 25, y el intervalo (rango de trabajo) de esta función es [-3,3].

PRUEBA	Inicial	Final	n
1	-3	3	5
2	-3	3	10
3	-3	3	15
4	-3	3	20
5	-3	3	25

Tabla 3.3. Datos de las pruebas para medir la ortogonalidad de los regresores en la función 3.10.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	3.00	3.775e-16	7.948e+15
2	5.50	2.808e-15	1.959e+15
3	8.00	5.999e-15	1.334e+15
4	10.50	2.903e-15	3.617e+15
5	13	7.212e-15	1.807e+15

Tabla 3.4. Parámetro de medición de la ortogonalidad (ratio diagonal) en la función 3.10.

Gráficas:

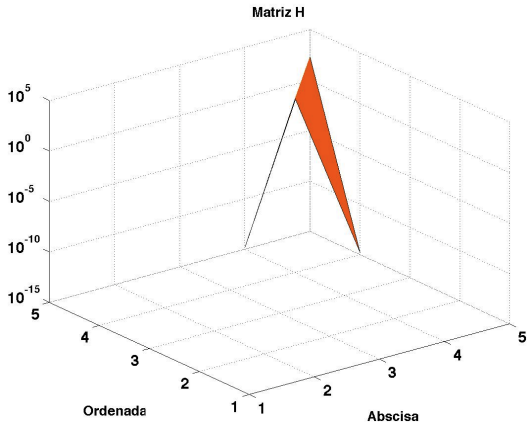


Figura3.9: Con 5 puntos de entrenamiento (prueba 1).

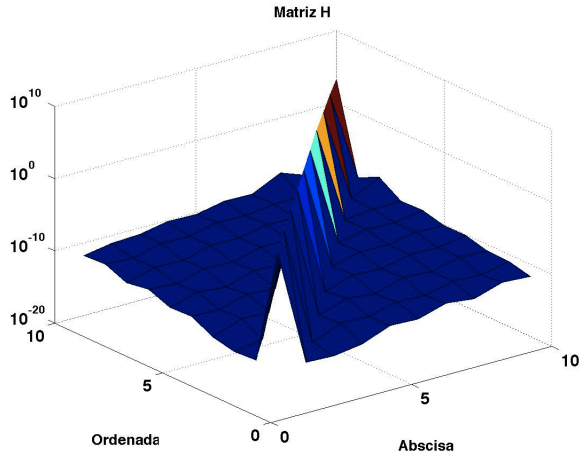


Figura3.10. Con 10 puntos de entrenamiento (prueba 2).

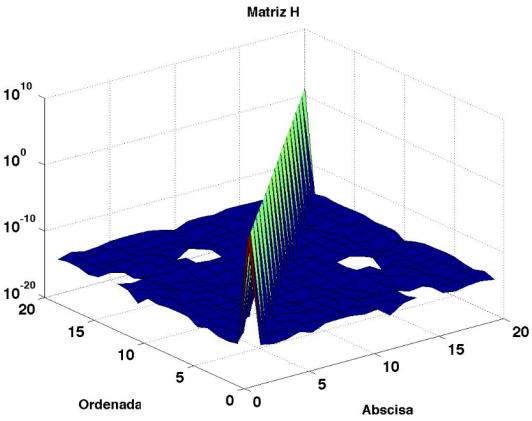


Figura3.11. Con 20 puntos de entrenamiento (prueba 4).

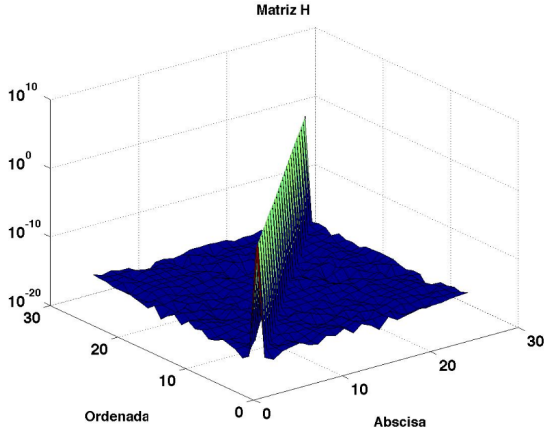


Figura3.12. Con 25 puntos de entrenamiento (prueba 5).

Tal como se observa en las figuras anteriores (3.9 a la 3.12) y en la tabla 3.4, vemos que la matriz **H** es diagonal y que por tanto los regresores son ortogonales. El valor del parámetro que hemos definido como ratio diagonal cuanto mayor sea, indica que la matriz **H** es mas diagonal. En la tabla 3.4 queda claro que el valor de ratio diagonal es elevado en todos los casos debido a que la media de la diagonal es distinta de cero y la media de los elementos fuera de la diagonal es prácticamente cero.

- Tabla y gráficas del estudio de la función 3.11

En la tabla 3.5 se muestran los datos para las pruebas para medir la ortogonalidad de los regresores en la función 3.11. El número de puntos de entrenamiento n, varía entre 5 y 25, y el intervalo (rango de trabajo) de esta función es [0,0.3].

PRUEBA	Inicial	Final	n
1	0	0.3	5
2	0	0.3	10
3	0	0.3	15
4	0	0.3	20
5	0	0.3	25

Tabla 3.5. Datos de las pruebas para medir la ortogonalidad de los regresores en la función 3.11.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	3.00	3.775e-16	7.948e+15
2	5.50	2.808e-15	1.959e+15
3	8.00	5.999e-15	1.334e+15
4	10.50	2.903e-15	3.617e+15
5	13	7.212e-15	1.803e+15

Tabla 3.6. Parámetro de medición de la ortogonalidad (ratio diagonal) en la función 3.11.

Gráficas:

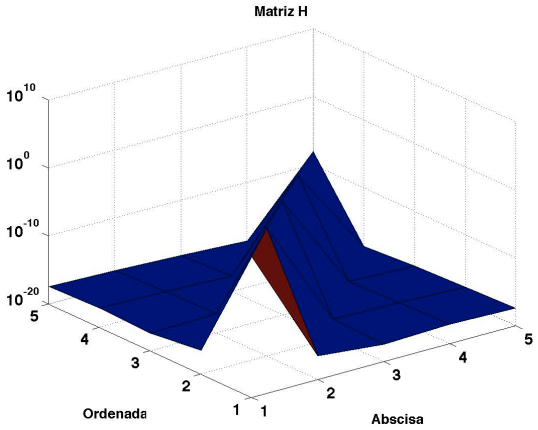


Figura3.13. Con 5 puntos de entrenamiento (prueba 1).

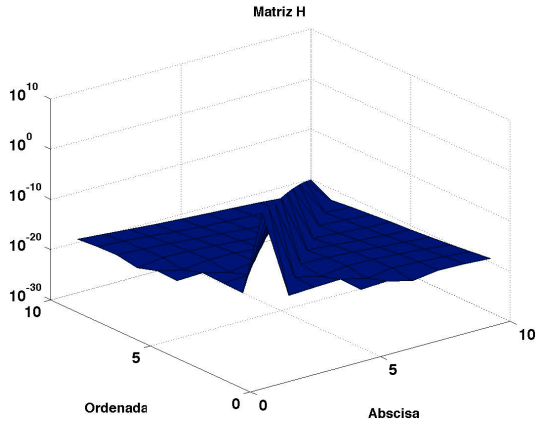


Figura3.14. Con 10 puntos de entrenamiento (prueba 2).

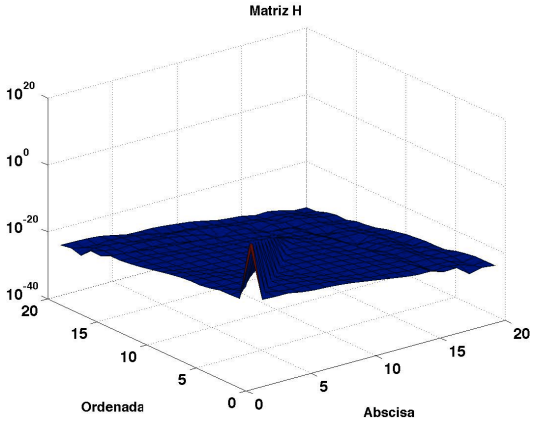


Figura3.15. Con 20 puntos de entrenamiento (prueba 4).

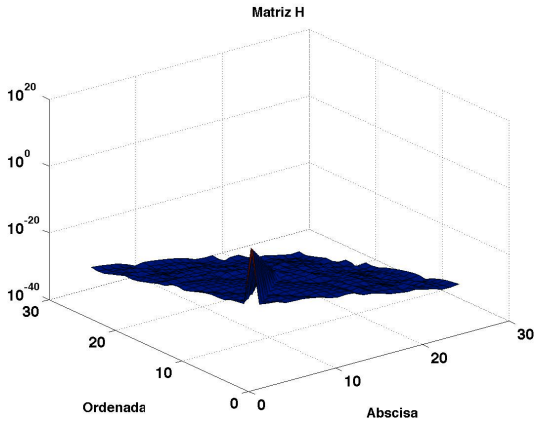


Figura3.16. Con 25 puntos de entrenamiento (prueba 5).

Tal como se observa en las figuras anteriores (3.13 a la 3.16) y en la tabla 3.6, vemos que la matriz **H** es diagonal y que por tanto los regresores son ortogonales. El valor del parámetro que hemos definido como ratio diagonal cuanto mayor sea, indica que la matriz **H** es mas diagonal. En la tabla 3.6 queda claro que el valor de ratio diagonal es elevado en todos los casos debido a que la media de la diagonal es distinta de cero y la media de los elementos fuera de la diagonal es prácticamente cero, por lo que los regresores son ortogonales. Su diagonal nos da números elevados y su media no diagonal se aproxima cada vez mas a cero.

3.2.2. Estudio del número de puntos de entrenamiento necesarios.

En este apartado vamos a estudiar cómo afecta el número de puntos de entrenamiento en la precisión de la aproximación. En este estudio, formamos una red neuronal en cada caso con todos los polinomios disponibles, es decir, el número de polinomios será n (número de puntos de entrenamiento). La precisión de la aproximación se mostrará de forma cuantitativa y cualitativa, tanto en los puntos de entrenamiento como en los de validación.

Los puntos de validación sirven para comprobar si la red generaliza correctamente, es decir, si la red aproxima bien la función deseada en unos puntos diferentes a los de entrenamiento. Los puntos de validación se han seleccionado de forma regular, son puntos equidistantes unos de otros. Además, se ha tomado un numero de puntos de validación (n val) elevado para comprobar la precisión de la red de forma exhaustiva .

Vamos a observar que cuando aumentamos el número de puntos de entrenamiento la aproximación mejora. Mejora el error de aproximación en los datos de entrenamiento y de validación. Además, la matriz de regresores seguirá siendo ortogonal aunque aumentemos el número de puntos de entrenamiento.

- Tabla y gráficas del estudio de la función 3.9

En la tabla 3.7 se muestran los datos para las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.9. El número de puntos de entrenamiento n, varía entre 5 y 10, y el intervalo (rango de trabajo) de esta función es [-2,2].

PRUEBA	Inicial	Final	n	n val
1	-2	2	**5**	100
2	-2	2	**7**	100
3	-2	2	**10**	100

Tabla 3.7: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.9.

Para medir de forma cuantitativa la precisión de la red neuronal hemos utilizado un parámetro de medición del error que se denomina FVU.

La ecuación del error FVU es:

$$FVU = \sqrt{\frac{\sum_{i=1}^{p}\left[y_i - F_i\right]^2}{\sum_{i=1}^{n}\left[y_i - \bar{y}\right]^2}} \qquad (3.12)$$

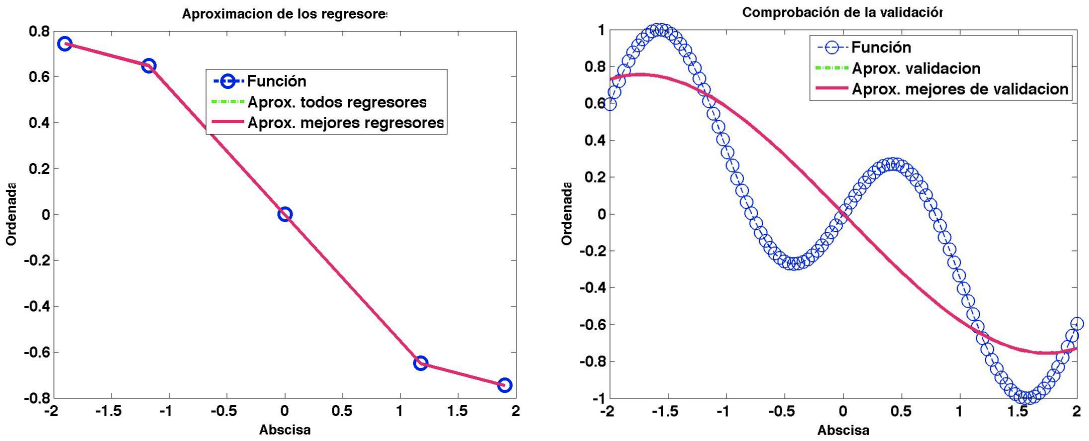
Ésta se ha empleado en diversos artículos. El parámetro "p" es una variable que nos indica el número de puntos de entrenamiento o de validación, "y" es el valor de la función deseada, "F" valor de salida de la red neuronal, e "$\bar{y}$ " es la media de los valores de *i* .

PRUEBA	n	Error FVU entrenamiento	Error FVU validación
1	**5**	1.895e-32	3.068e-01
2	**7**	3.591e-32	2.967e-02
3	**10**	5.496e-32	9.013e-06

Tabla 3.8: Parámetros de error de entrenamiento y validación de la función 3.9.

Gráficas:

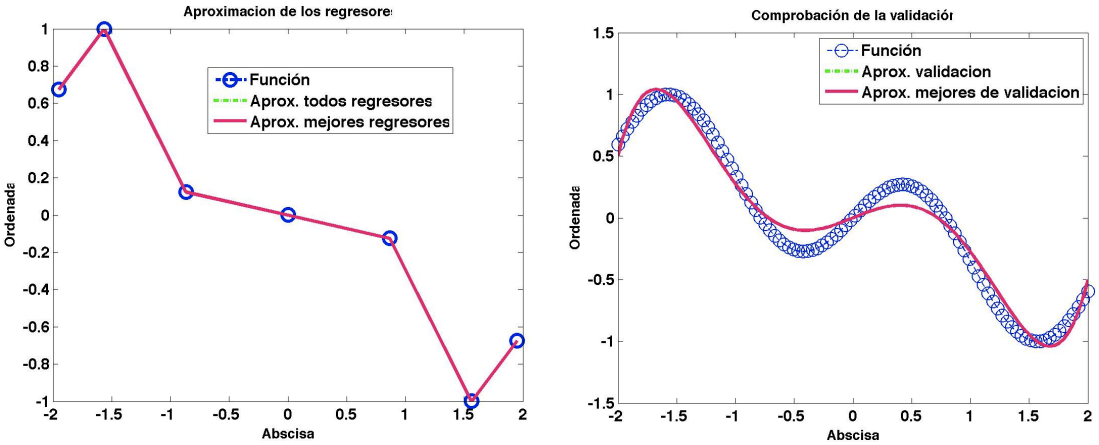
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 3.17: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

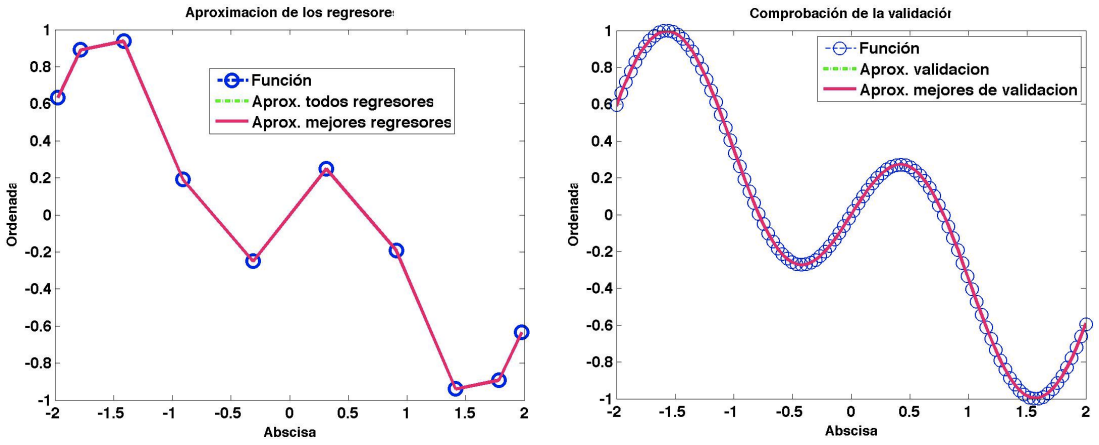
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 3.18: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

* Diez puntos de entrenamiento.



(a) **(b)**

Figura 3.19: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

Como observamos en las gráficas anteriores, con 5 puntos de entrenamiento la función no está bien muestreada y la salida de la red neuronal se parece a una recta; con 7 puntos de entrenamiento la función es mejor muestreada y con 10 puntos de entrenamiento el muestreo es suficiente para conseguir una buena aproximación tanto de entrenamiento como de validación.

- Tabla y gráficas del estudio de la función 3.10

PRUEBA	Inicial	Final	n	n val
1	-3	3	**5**	100
2	-3	3	**7**	100
3	-3	3	**15**	100

Tabla 3.9: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.10.

Tabla:

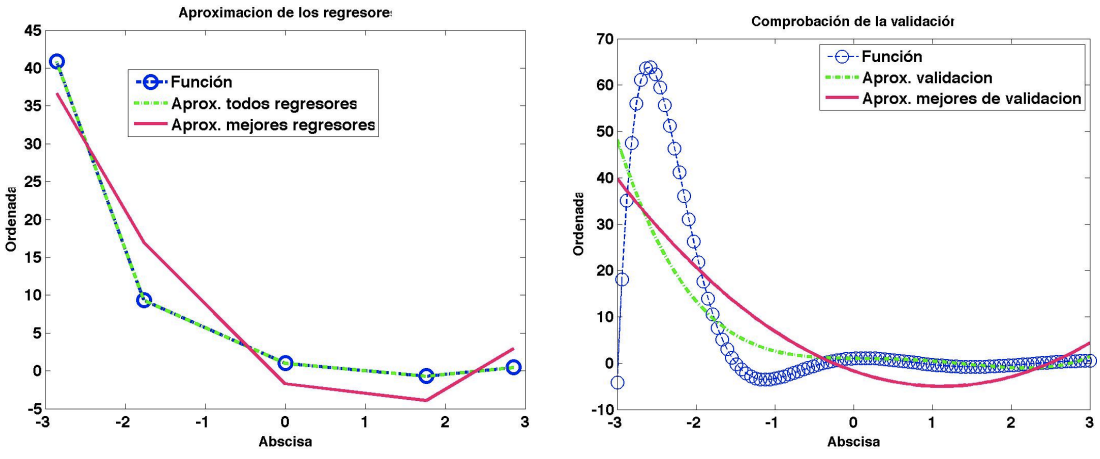
PRUEBA	n	Error FVU entrenamiento	Error FVU validación
1	**5**	7.961e-32	4.154e-01
2	**7**	3.109e-32	7.077e-02
3	**15**	6.642e-32	2.868e-09

Tabla 3.10: Parámetros de error de entrenamiento y validación de la función 3.10.

En la tabla 3.10, el error es muy pequeño y se reduce en gran medida con el aumento de los puntos de entrenamiento.

Gráficas:

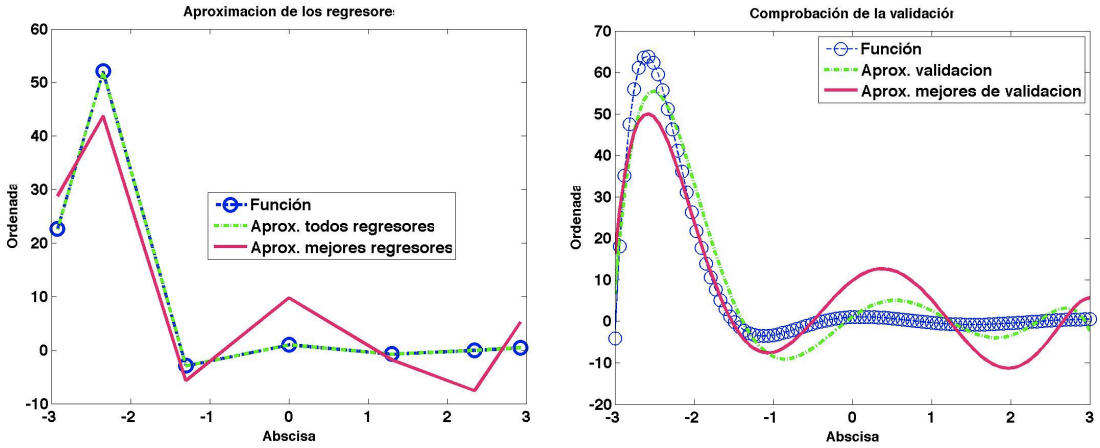
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 3.20: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

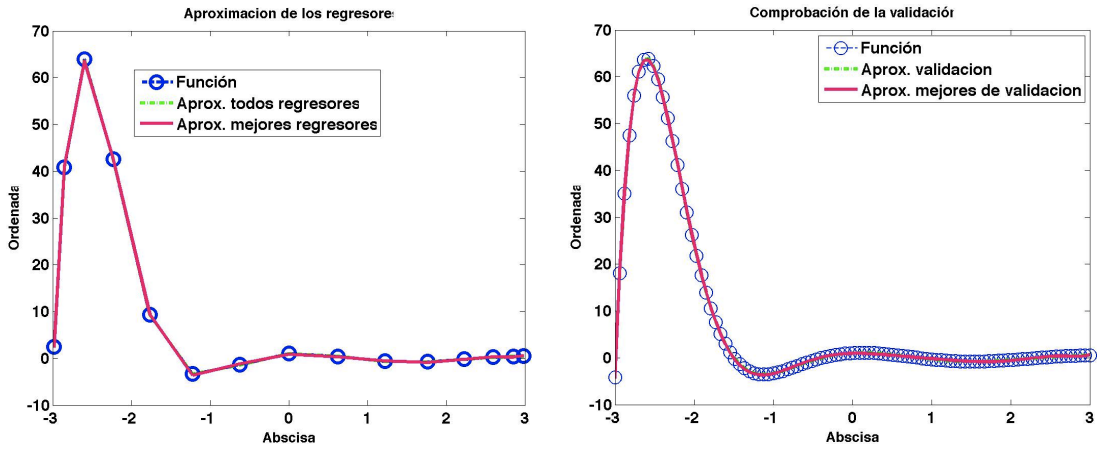
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 3.21: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

* Quince puntos de entrenamiento.



(a) **(b)**

Figura 3.22: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

Como observamos en las gráficas anteriores, con 5 puntos de entrenamiento la función no se muestrea adecuadamente, no tenemos suficiente información de cómo cambia la función, especialmente cerca de -3, donde la función cambia rápidamente. Ésta es una función difícil de aproximar. En cambio, cuando tenemos 7 puntos de entrenamiento la función se representa mucho mejor. La aproximación mejora. Cuando tenemos 15 puntos de entrenamiento la aproximación tanto en los datos de entrenamiento como sobre todo en los datos de validación es muy precisa.

- Tabla y gráficas del estudio de la función 3.11

PRUEBA	Inicial	Final	n	n val
1	0	0.3	**5**	100
2	0	0.3	**7**	100
3	0	0.3	**10**	100
4	0	0.3	**15**	100

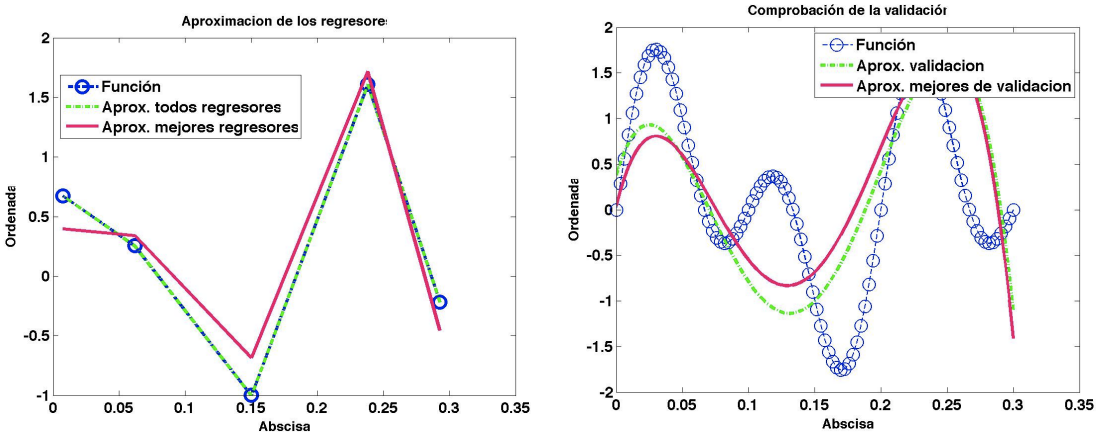
Tabla 3.11: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.11.

PRUEBA	n	Error FVU entrenamiento	Error FVU validación
1	**5**	7.310e-32	7.040330e-01
2	**7**	3.432e-32	7.211e-01
3	**10**	1.241e-31	2.745e-02
4	**15**	7.742e-32	1.241e-05

Tabla 3.12: Parámetros de error de entrenamiento y validación de la función 3.11.

Gráficas:

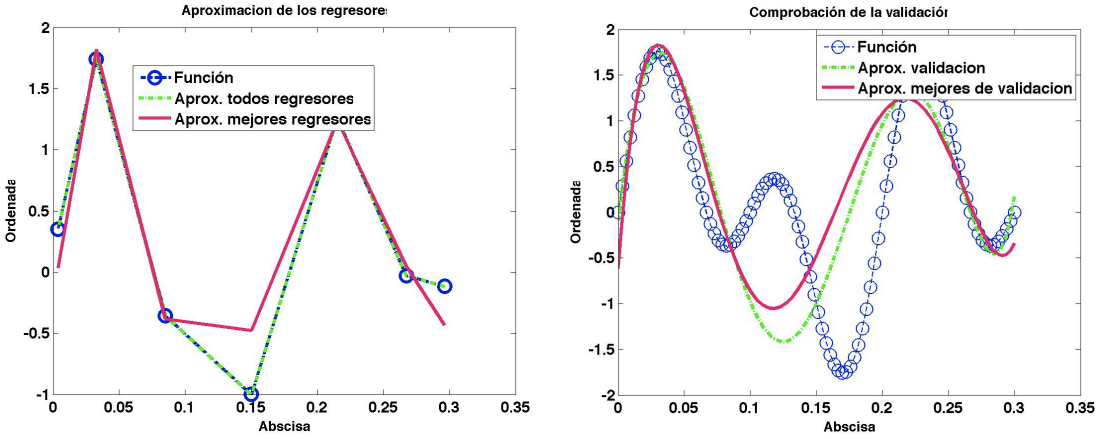
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 3.23: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

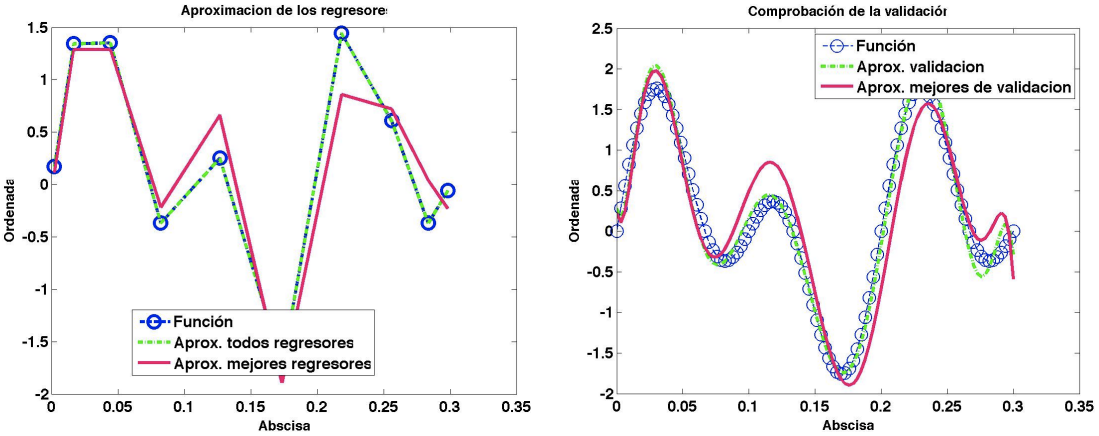
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 3.24: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

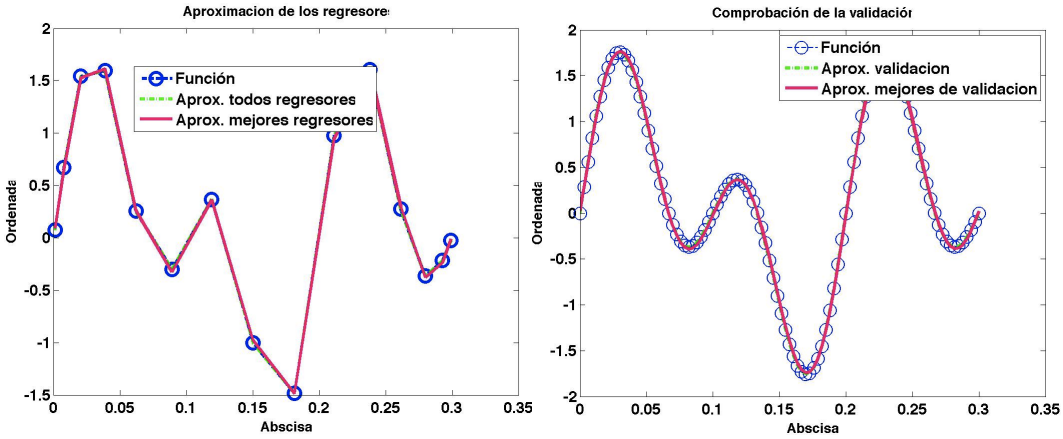
* Diez puntos de entrenamiento.



(a) **(b)**

Figura 3.25: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

* Quince puntos de entrenamiento.



(a) **(b)**

Figura 3.26: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=15 (prueba 4).

Ésta función presenta cambios y oscilaciones en el rango estudiado[0, 0.3] que hacen necesario por lo menos 10 puntos de entrenamiento para una representación mínima de cómo es la función.

Podemos observar que es necesario un numero mínimo de puntos de entrenamiento para conseguir una representación adecuada de la función. Sin esa adecuada representación no será posible que la red ante una entrada nueva proporcione una salida correcta.

Las redes de Chebychev, al emplear puntos de entrenamiento que son ceros de Chebychev, no posee un conjunto de puntos de entrenamiento equi-espaciados ya que los ceros de Chebychev no se distribuyen de forma regular. Conforme aumenta el número de puntos, éstos aparecen en mayor medida en los extremos del intervalo de trabajo.

Un ejemplo muy visible seria en la función 3.11, dónde los puntos de entrenamiento tienden a agruparse en los extremos, sin embargo, como tenemos ya muchos puntos de entrenamiento la aproximación de validación es buena.

Datos:

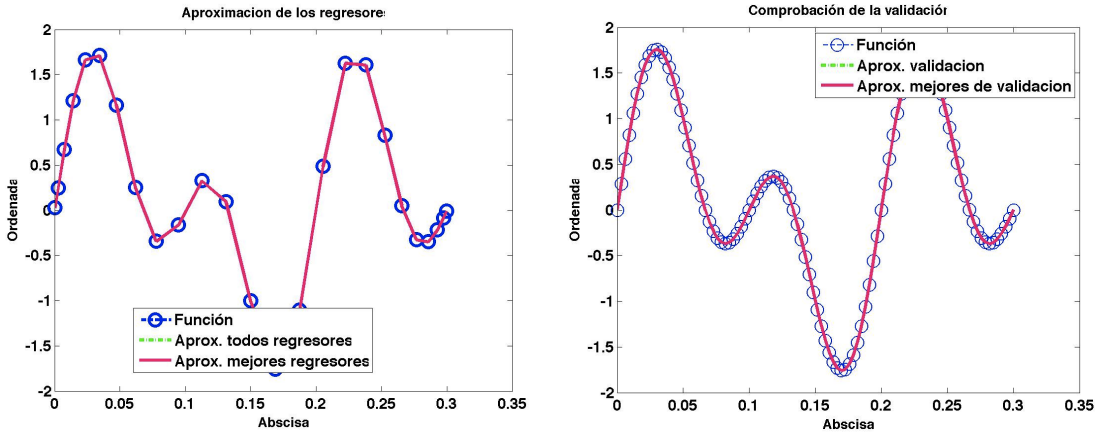
PRUEBA	Inicial	Final	n	n val
5	0	0.3	25	100

Tabla 3.13: Datos de la prueba para estudiar el número de puntos de entrenamiento necesarios para la función 3.11.

PRUEBA	Error FVU entrenamiento	Error FVU validación
5	9.595e-32	7.458e-18

Tabla 3.14: Parámetros de error de entrenamiento y validación de la función 3.11.

Gráficas:



(a) **(b)**

Figura3.27: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=25 (prueba 5).

3.2.3. Estudio del número de regresores necesarios.

En este apartado estudiaremos qué ocurre cuando seleccionamos un subconjunto de polinomios para formar la red neuronal del total de polinomios disponibles. En particular, se estudiará cómo evoluciona el error de aproximación cuando aumentamos progresivamente el número de polinomios seleccionados. En todas las pruebas, el número de puntos de entrenamiento es constante, y por tanto, el número de polinomios es constante.

- Tabla y gráficas del estudio de la función 3.9

En la siguiente tabla se muestran los datos que hemos utilizado para estudiar el número de polinomios necesarios para la aproximación de la función 3.9.

En esta tabla 3.15, número neuronas mejores, indica el número de neuronas (polinomios) que forman la red neuronal. Si por ejemplo el número de neuronas es 3, se utilizan tres polinomios de entre los polinomios de orden 0 hasta 6 (n=7).

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	-2	2	7	100	**1**
2	-2	2	7	100	**2**
3	-2	2	7	100	**3**
4	-2	2	7	100	**4**

Tabla 3.15 : Datos de la prueba para estudiar el número de regresores necesarios para la función 3.9.

Para medir la calidad de la aproximación de las redes neuronales estudiadas, recurrimos de nuevo al estudio cualitativo mediante gráficas, y cuantitativo mediante el error FVU. Además, como se observa en la tabla 3.16, también medimos la calidad de aproximación mediante un parámetro que denominamos porcentaje de energía; éste parámetro mide el cociente de la energía de la señal deseada, expresada según la ecuación (2.16) y la energía de los valores de salida de la red neuronal. Hay que tener en cuenta que el valor máximo de porcentaje de energía es uno.

PRUEBA	Error FVU entrenamiento (mejores neuronas)	Error FVU validación (mejores neuronas)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	1.329e-01	2.388734e-01	1	0.867
2	3.132e-04	2.993e-02	1	0.999
3	2.180e-31	2.967e-02	1	1
4	1.624e-31	2.967e-02	1	1

Tabla 3.16 : Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.9.

Gráficas:

A continuación vamos a mostrar las gráficas de aproximación para el conjunto de puntos de validación y la contribución de cada polinomio a la aproximación de la función deseada. Ésta contribución se muestra en la figura 3.28, e indica para cada polinomio el valor del parámetro “err” de la ecuación 2.24. Como se explicó en el

capítulo 2, cuanto mayor sea este parámetro, mejor será el polinomio en la aproximación.

Como se aprecia en la figura 3.28, el mejor polinomio es el segundo , que posee orden 1. El segundo mejor polinomio es el sexto que posee orden 5.El resto de polinomios apenas contribuyen a la mejora de la aproximación deseada. Como vemos en la tabla 3.15, la utilización de más de dos polinomios no mejora significativamente la aproximación. Con dos polinomios el porcentaje de energía es ya prácticamente del cien por cien. En las gráficas donde se muestra la aproximación de validación, también se observa este hecho. La aproximación con las dos ultimas gráficas, es prácticamente la misma que la segunda gráfica. De la primera gráfica a la segunda, si se nota un cambio apreciable ya que pasamos de realizar una aproximación con un polinomio que es una recta (aproximación lineal), a tener dos polinomios que permite una aproximación no lineal.

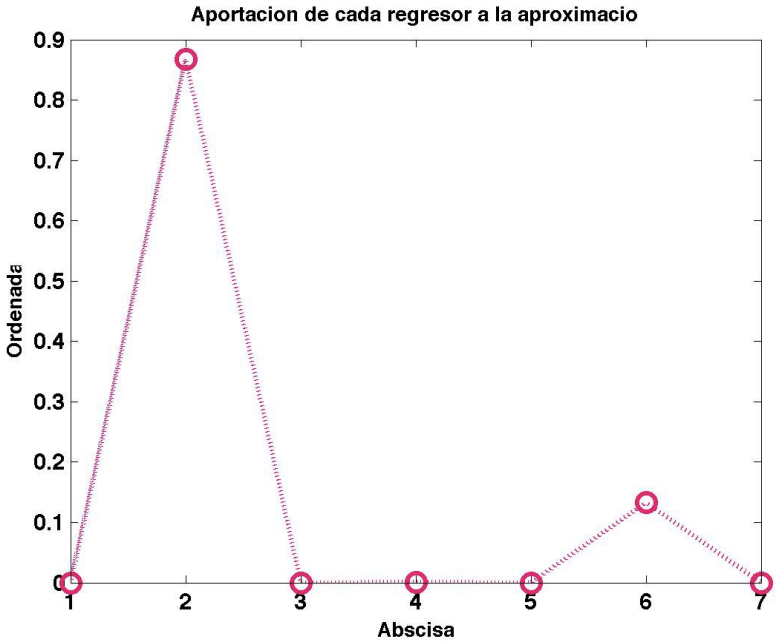


Figura 3.28: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 7, y la ordenada es la contribución a la salida (prueba 1).

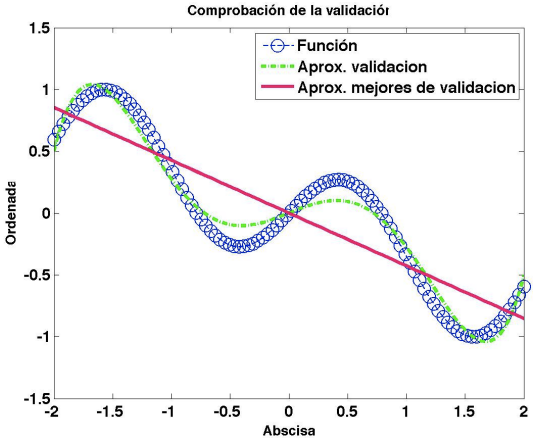


Figura3.29: con una mejor neurona (prueba 1).

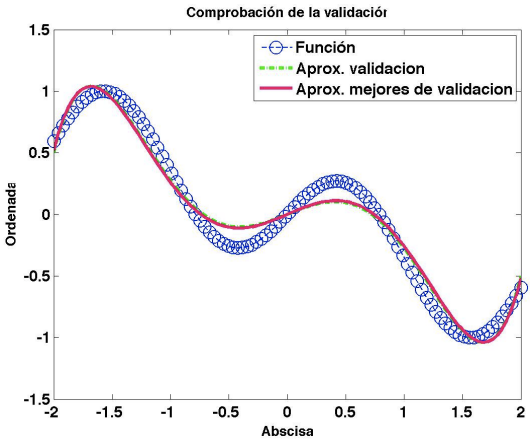


Figura3.30: con dos mejores neuronas (prueba 2).

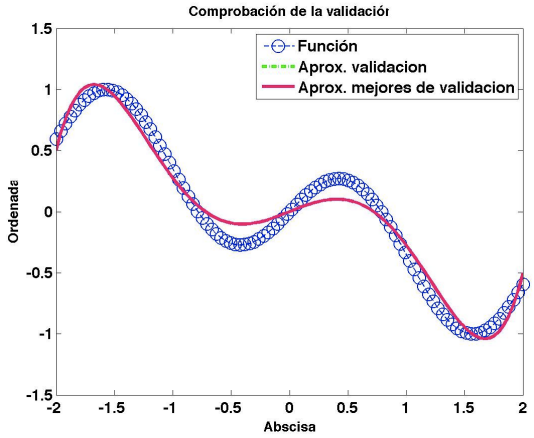


Figura3.31: con tres mejores neuronas (prueba 3).

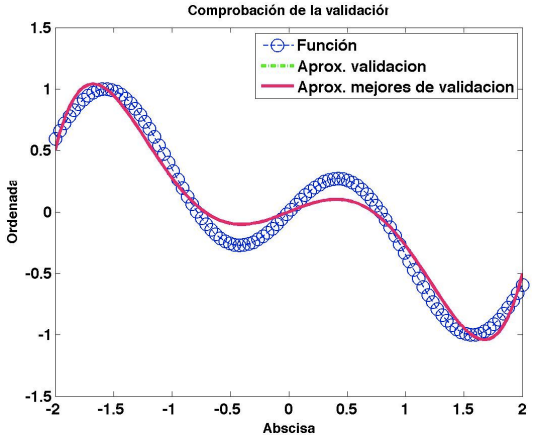


Figura3.32: con cuatro mejores neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.10

A continuación de muestran los datos de la aproximación de la función (3.10). Ésta función es mas difícil de aproximar, y por ello hemos elegido doce puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	-2	2	12	100	**1**
2	-2	2	12	100	**3**
3	-2	2	12	100	**5**
4	-2	2	12	100	**7**

Tabla 3.17 : Datos de la prueba para estudiar el número de regresores necesarios para la función 3.10.

En la tabla 3.18, se muestra el error FVU y el porcentaje de energía para los datos de entrenamiento y validación.

PRUEBA	Error FVU entrenamiento (mejores neuronas)	Error FVU validación (mejores neuronas)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	7.968e-01	1.014e+00	1	0.299
2	2.844e-01	6.406e-01	1	0.749
3	1.537e-02	4.046e-02	1	0.986
4	8.546e-05	1.899e-04	1	0.999

Tabla 3.18 : Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.10.

Gráficas:

De nuevo el mejor polinomio es de orden uno, que es el segundo polinomio. Luego tenemos 5 polinomios que aportan bastante a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores(polinomios), la aproximación va a ir mejorando. En la función anterior, el error caía bruscamente de tener un polinomio a dos y luego prácticamente no descendía. En cambio, ahora el

error de aproximación va a ir descendiendo constantemente. Efectivamente, este hecho se constata con los datos de la tabla3.18 y las gráficas de la aproximación de validación de la prueba 1 a la 4.

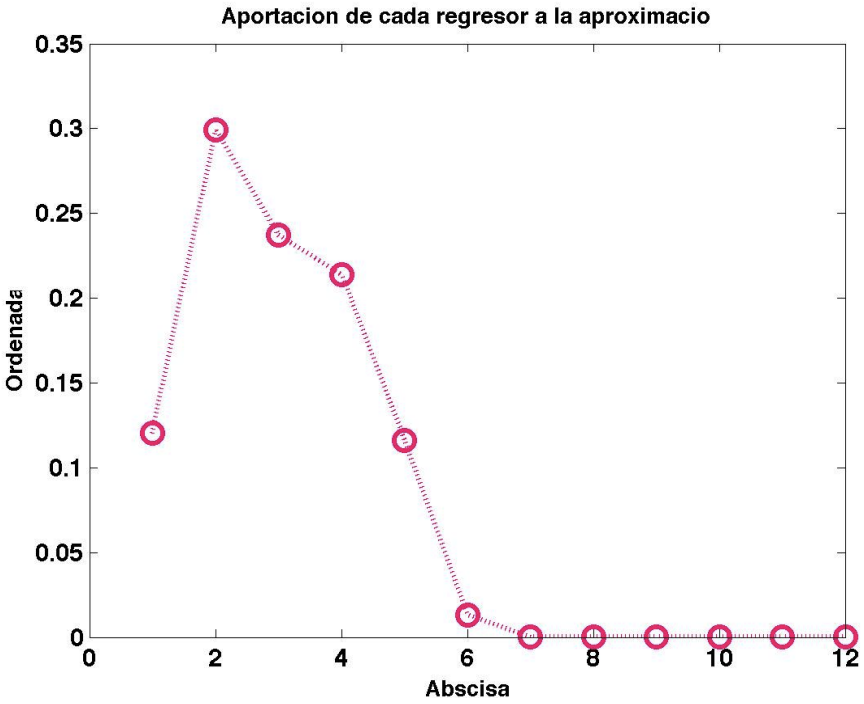


Figura3.33: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 12, y la ordenada es la contribución a la salida.

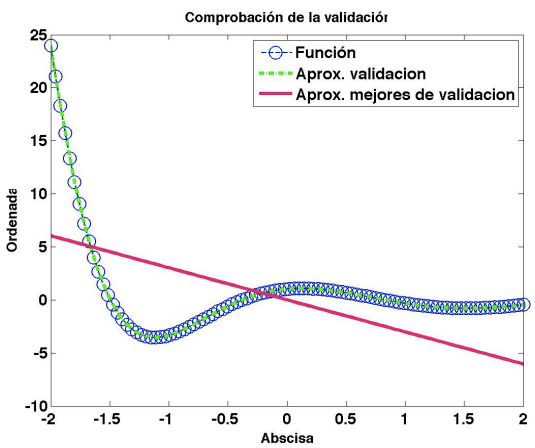


Figura3.34: con una mejor neurona (prueba 1).

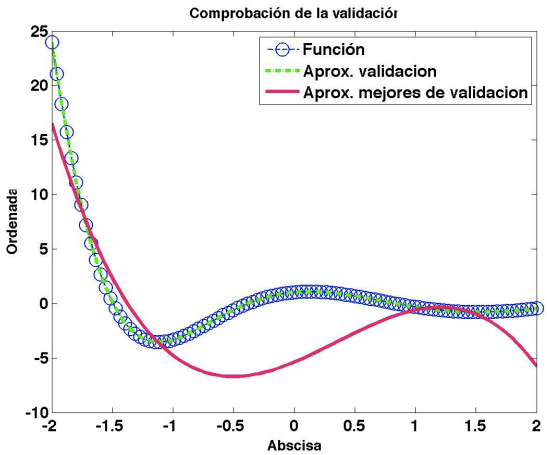


Figura3.35: con tres mejores neuronas (prueba 2).

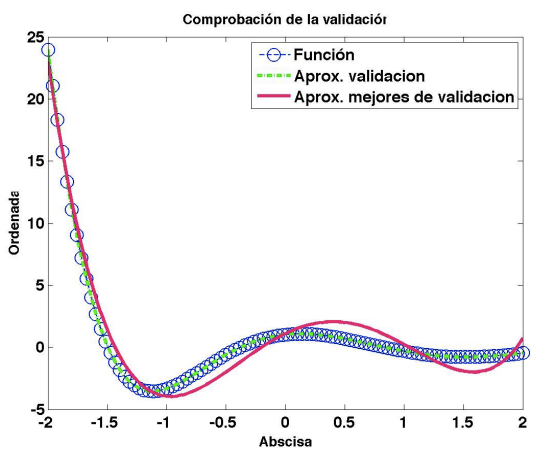


Figura3.36: con cinco mejores neuronas (prueba 3).

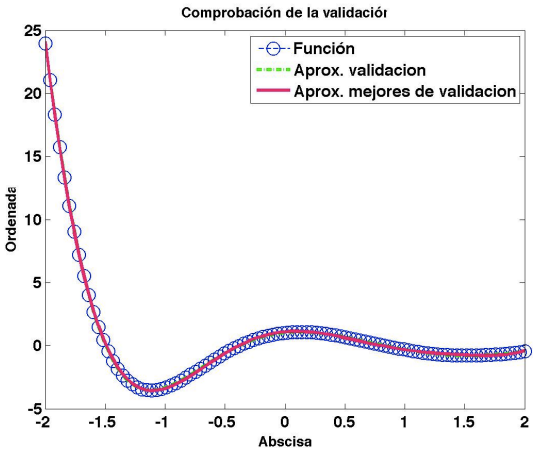


Figura3.37: con siete mejores neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.11

A continuación de muestran los datos de la aproximación de la función (3.11). Ésta función también es difícil de aproximar, y por ello hemos elegido quince puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	0	0.3	15	100	**1**
2	0	0.3	15	100	**4**
3	0	0.3	15	100	**7**
4	0	0.3	15	100	**10**

Tabla 3.19: Datos de la prueba para estudiar el número de regresores necesarios para la función 3.11.

PRUEBA	Error FVU entrenamiento (mejores neuronas)	Error FVU validación (mejores neuronas)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	7.368e-01	6.071e-01	1	0.325
2	2.572e-01	1.876e-01	1	0.764
3	6.534e-02	5.713e-02	1	0.940
4	1.118e-03	1.023e-03	1	0.999

Tabla 3.20 : Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.11.

Gráficas:

De nuevo el mejor polinomio es de orden cuatro, que es el quinto polinomio. Luego tenemos 6 polinomios que aportan bastante a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores(polinomios), la aproximación va a ir mejorando. En la función 3.9, el error caía bruscamente de tener un polinomio a dos y luego prácticamente no descendía. En cambio, ahora el error de aproximación va a ir descendiendo constantemente como ocurría en la función 3.10. Efectivamente, este hecho se constata con los datos de la tabla 3.20 y las gráficas de la aproximación de validación de la prueba 1 a la 4.

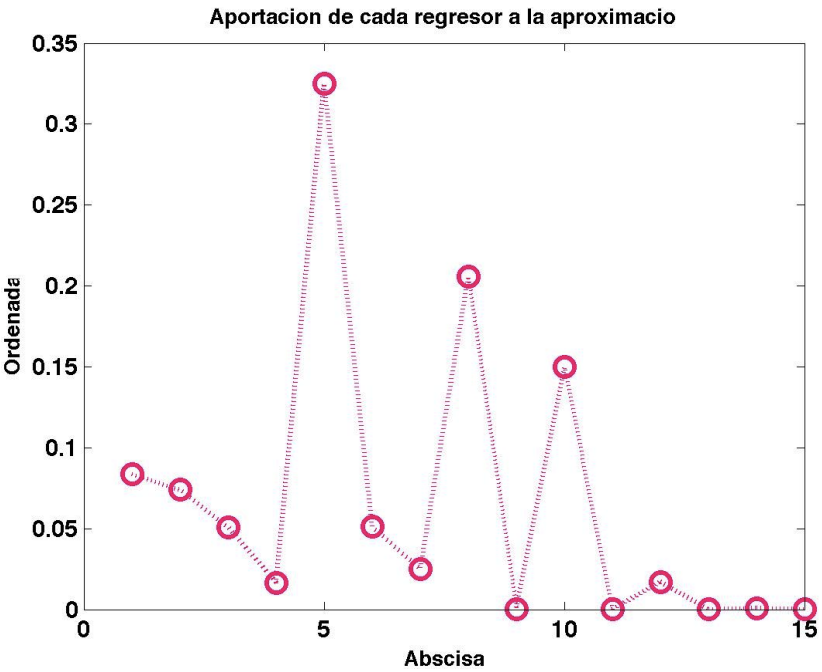


Figura3.38: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 15, y la ordenada es la contribución a la salida (prueba 1).

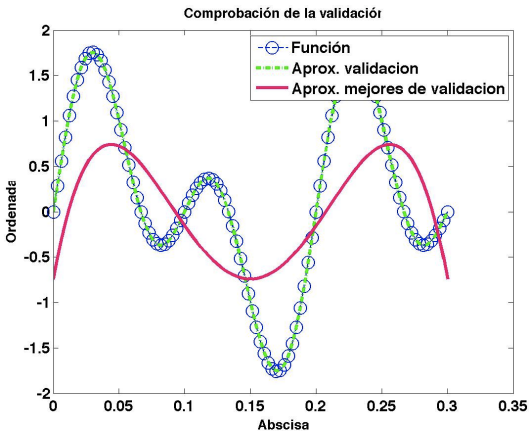


Figura3.39: con una mejor neurona (prueba 1).

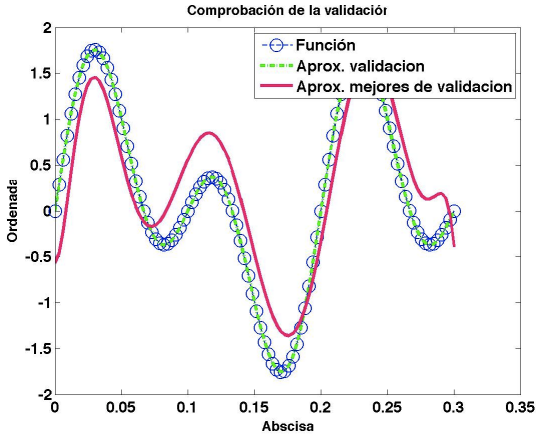


Figura3.40: con cuatro mejores neuronas (prueba 2).

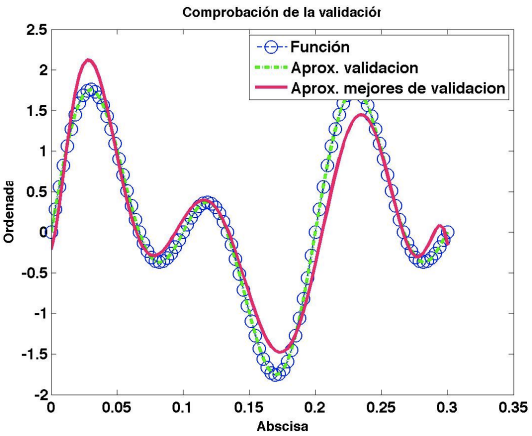


Figura3.41: con siete mejores neuronas (prueba 3).

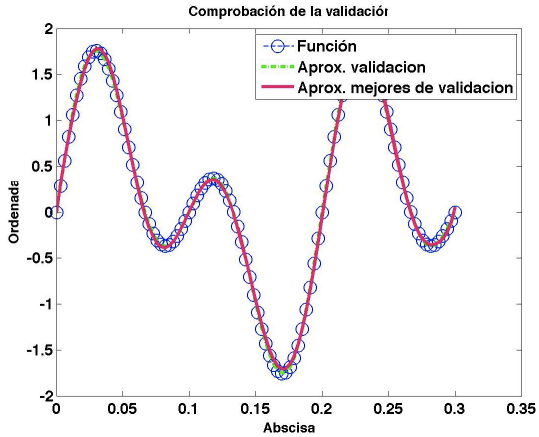


Figura3.42: con diez mejores neuronas (prueba 4).

Como conclusión al estudio de este apartado, podemos decir que conforme aumentamos el número de regresores vemos cómo mejora la aproximación. Los polinomios de Chebychev de primera especie, demuestran su capacidad de aproximación de diversas funciones en diferentes intervalos. El algoritmo OLS permite seleccionar los mejores polinomios de forma independiente. De este modo, en los casos estudiados se consigue una red neuronal muy pequeña que alcanza una elevada precisión en la tarea de aproximación.

3.2.4. Estudio del comportamiento de la red neuronal en presencia de ruido en los datos de entrenamiento.

Para finalizar este capítulo, estudiamos qué ocurre en la aproximación cuando los datos de entrenamiento están contaminados con ruido. Para ello mostraremos el error de aproximación y las gráficas de entrenamiento y validación para diferentes número de puntos de entrenamiento, de regresores y diferentes niveles de ruido (SNR). Hay que destacar que el ruido es de tipo blanco gaussiano.

- Tabla y gráficas del estudio de la función 3.9

La tabla 3.21 se muestran los datos de las pruebas realizadas en la función 3.9 cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	-2	2	7	100	3	**5**
2	-2	2	7	100	3	**10**
3	-2	2	7	100	3	**15**
4	-2	2	7	100	3	**20**
5	-2	2	7	100	5	**5**
6	-2	2	7	100	5	**10**
7	-2	2	7	100	5	**15**
8	-2	2	7	100	5	**20**
9	-2	2	12	100	3	**5**
10	-2	2	12	100	3	**10**
11	-2	2	12	100	3	**15**
12	-2	2	12	100	3	**20**
13	-2	2	12	100	5	**5**
14	-2	2	12	100	5	**10**
15	-2	2	12	100	5	**15**
16	-2	2	12	100	5	**20**

Tabla 3.21: Estudio de la función 3.9 cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

A continuación, mostramos los errores de aproximación y validación para las pruebas anteriores. Las dos primeras columnas corresponden a la red neuronal compuesta con todos los posibles polinomios, y las dos últimas columnas, a la red neuronal que está compuesta por las mejores neuronas.

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	4.568e-32	3.402e-01	1.087e-02	3.634e-01
2	7.532e-32	1.831e-01	5.877e-02	9.930e-02
3	7.932e-32	1.202e-01	1.886e-02	7.901e-02
4	5.508e-32	9.271e-02	2.296e-02	5.441e-02
5	9.150e-32	1.540e-01	1.389e-02	1.702e-01
6	3.112e-32	1.398e-01	1.229e-03	1.463e-01
7	5.051e-32	9.698e-02	1.382e-03	9.304e-02
8	8.945e-32	6.565e-02	1.137e-03	5.876e-02
9	6.972e-32	2.431e-01	2.103e-02	1.939e-01
10	1.900e-31	1.047e-01	7.112e-03	1.038e-01
11	5.547e-32	7.533e-02	4.727e-03	7.509e-02
12	5.405e-32	6.017e-02	4.822e-03	5.178e-02
13	7.728e-32	3.074e-01	4.736e-03	2.959e-01
14	1.315e-31	1.239e-01	1.627e-04	1.238e-01
15	8.841e-32	7.728e-02	2.639e-04	8.075e-02
16	3.413e-32	6.993e-02	3.002e-04	6.953e-02

Tabla 3.22 : Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.9.

. Del estudio de la tabla 3.22 , deducimos que la aproximación mejora cuando aumentamos el número de puntos de entrenamiento y que también mejora cuando aumentamos el número de neuronas en la red. Sin embargo, el error de aproximación en los datos de validación, nunca es muy reducido; esto es debido a que el ruido contamina la función y por lo tanto impide que la red neuronal aproxime correctamente la función deseada. Este hecho se ve mas claramente en las siguientes gráficas:

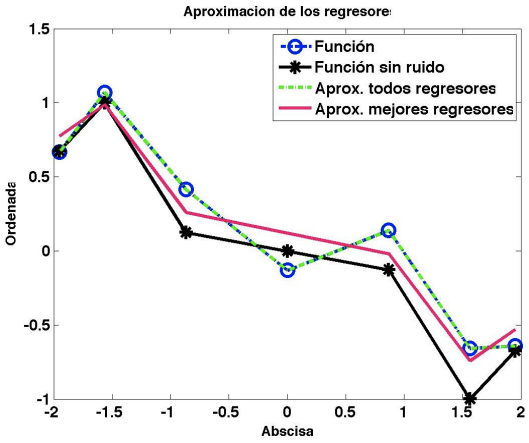


Figura3.43: aprox. entrenamiento con la función sin ruido (prueba 2)**.**

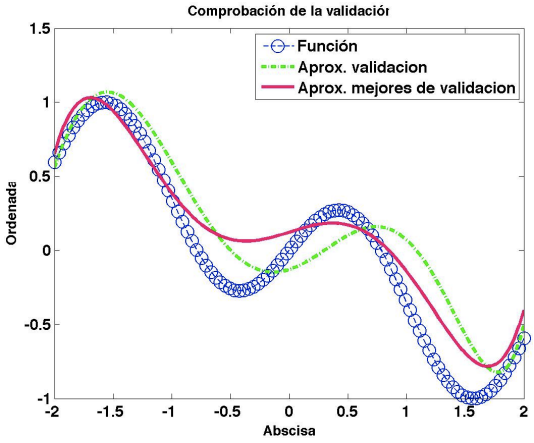


Figura3.44: aprox. validación (prueba 2)**.**

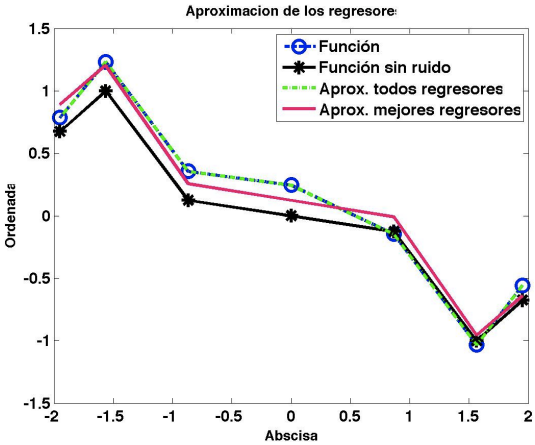


Figura3.45: aprox. entrenamiento con la función sin ruido (prueba 3)**.**

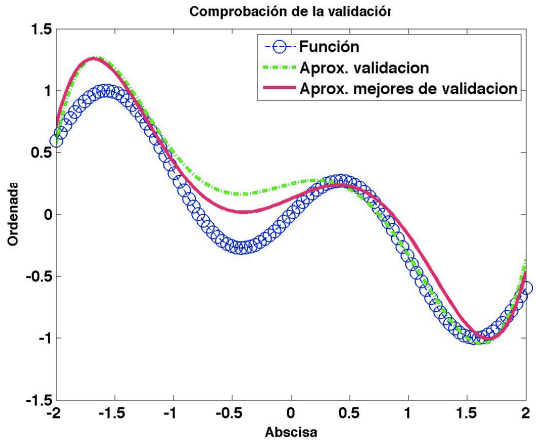


Figura3.46: aprox. validación (prueba 3)**.**

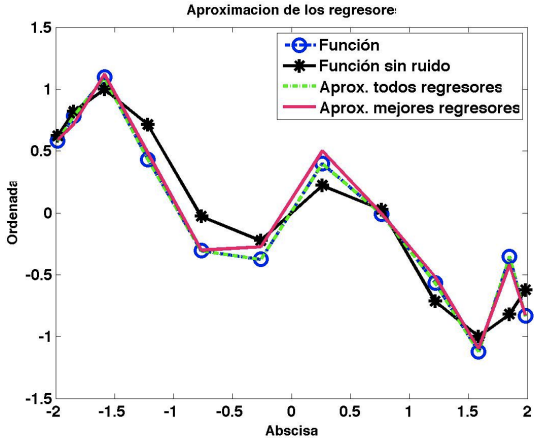


Figura3.47: aprox. entrenamiento con la función sin ruido (prueba 10)**.**

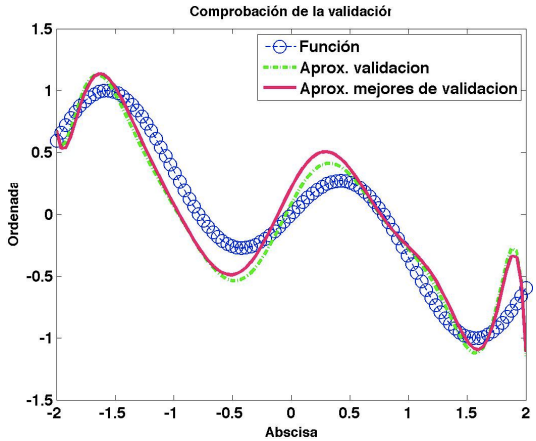


Figura3.48: aprox. validación (prueba 10)**.**

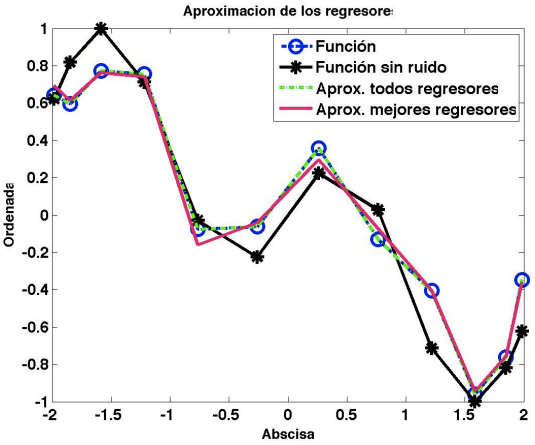


Figura3.49: aprox. entrenamiento con la función sin ruido (prueba 11)**.**

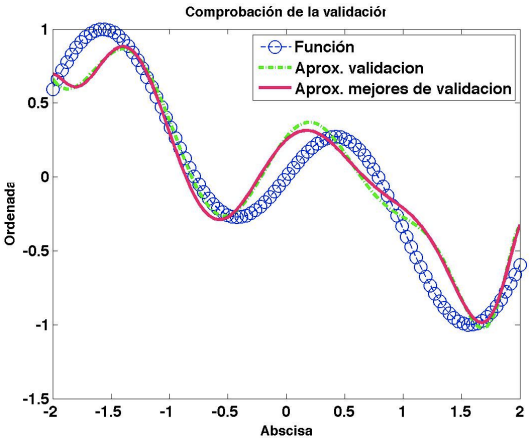


Figura3.50: aprox. validación (prueba 11)**.**

- Tabla y gráficas del estudio de la función 3.10

PRUEBA	Inicial	Final	n	n val	Numero mejores neuronas	SNR
1	-3	3	12	100	8	**10**
2	-3	3	12	100	8	**15**
3	-3	3	20	100	16	**10**
4	-3	3	20	100	16	**15**

Tabla 3.23: Estudio de la función 3.10 cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	1.044e-31	1.317e-01	1.176e-02	1.647e-01
2	1.109e-31	1.055e-01	5.922e-03	1.065e-01
3	7.848e-32	1.973e-01	3.164e-04	1.968e-01
4	6.269e-32	1.083e-01	3.174e-04	1.116e-01

Tabla 3.24 : Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.10.

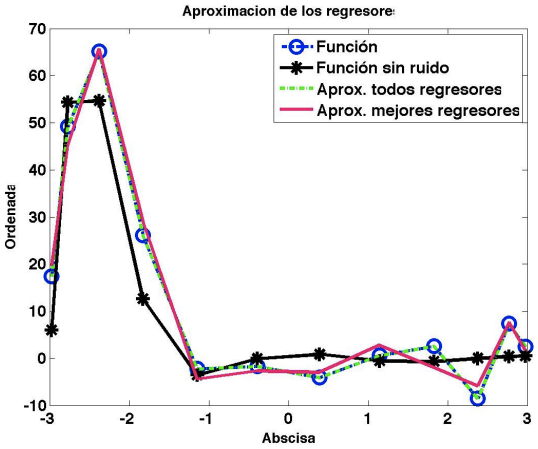


Figura3.51: aprox. entrenamiento con la función sin ruido (prueba 1)**.**

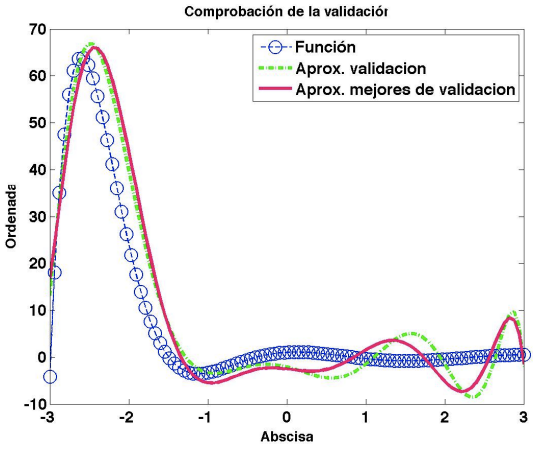


Figura3.52: aprox. validación (prueba 1)**.**

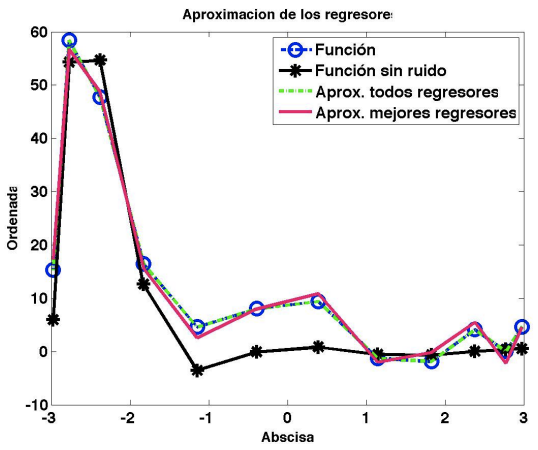


Figura3.53: aprox. entrenamiento con la función sin ruido (prueba 2)**.**

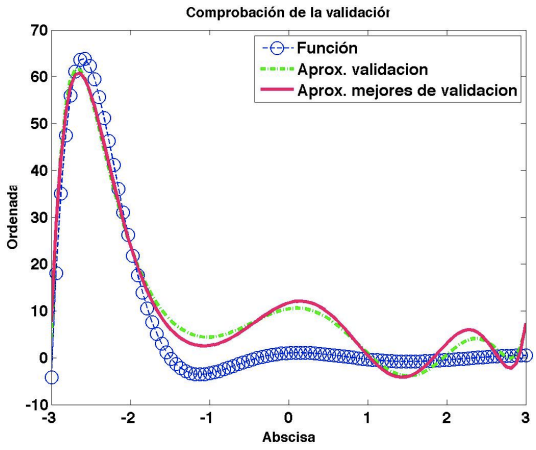


Figura3.54: aprox. validación (prueba 2)**.**

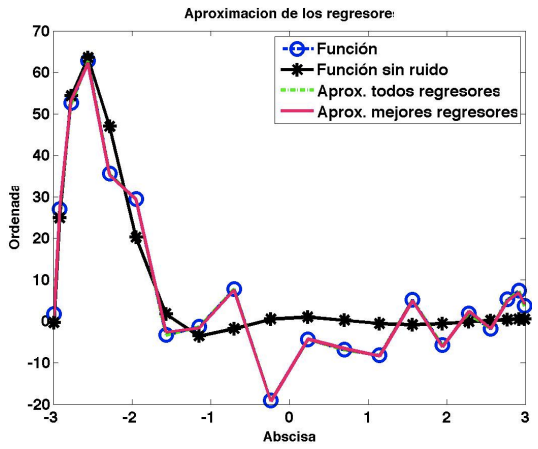


Figura3.55: aprox. entrenamiento con la función sin ruido (prueba 3)**.**

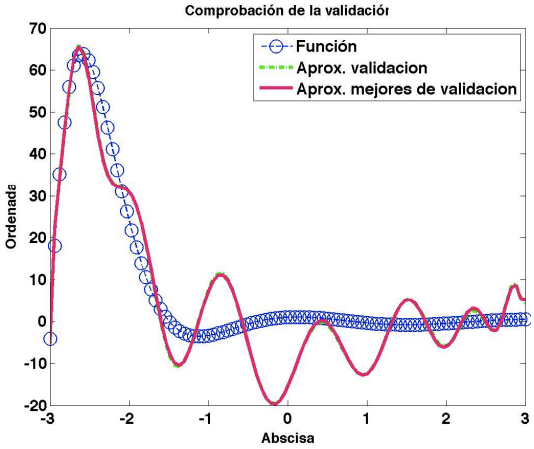


Figura3.56: aprox. validación (prueba 3)**.**

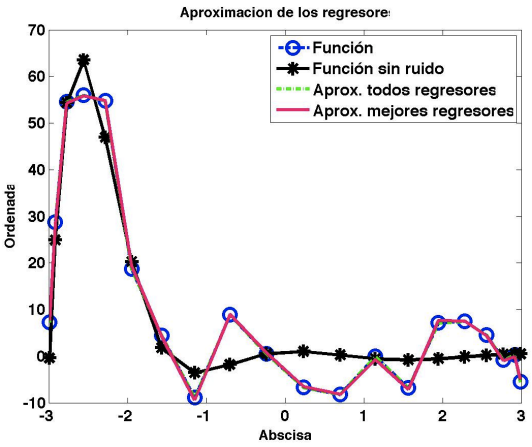


Figura3.57: aprox. entrenamiento con la función sin ruido (prueba 4)**.**

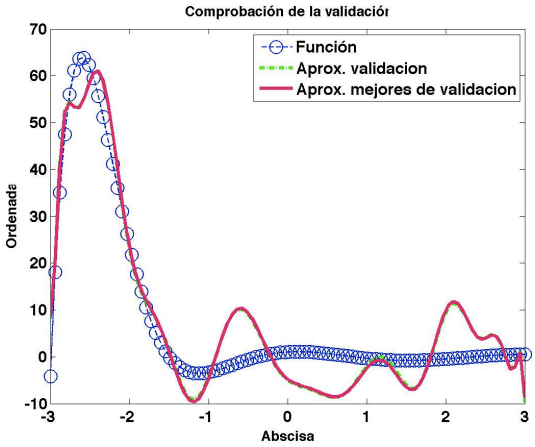


Figura3.58: aprox. validación (prueba 4)

- Tabla y gráficas del estudio de la función 3.11

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	0	0.3	12	100	8	**10**
2	0	0.3	12	100	8	**15**
3	0	0.3	20	100	16	**10**
4	0	0.3	20	100	16	**15**

Tabla 3.25: Estudio de la función 3.11 cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	6.291e-32	7.741e-02	2.586e-03	6.480e-02
2	8.972e-32	6.161e-02	4.366e-02	1.071e-01
3	8.444e-32	1.026e-01	3.349e-03	1.008e-01
4	9.735e-32	6.187e-02	9.103e-04	6.153e-02

Tabla 3.26: Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.11.

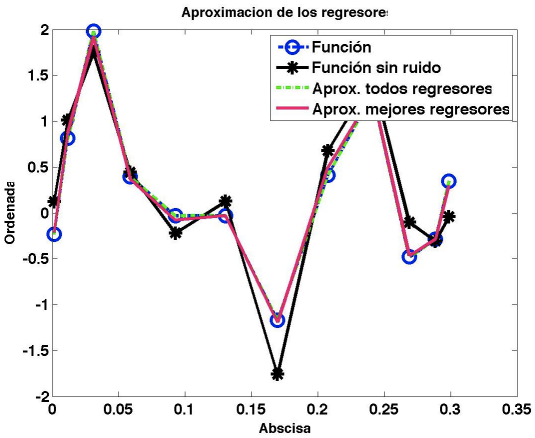


Figura3.59: aprox. entrenamiento con la función sin ruido (prueba 1).

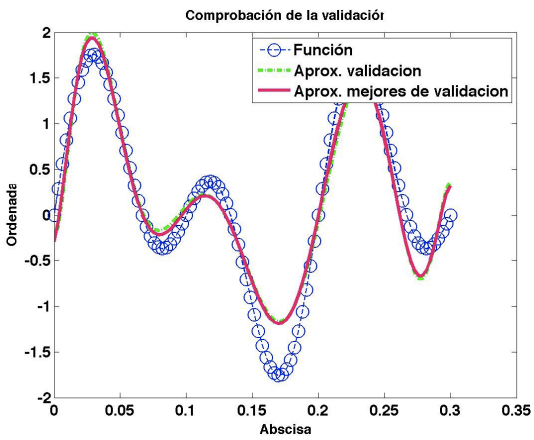


Figura3.60: aprox. validación (prueba 1).

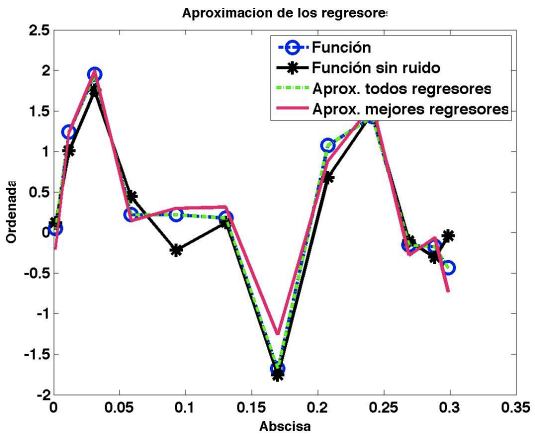


Figura3.61: aprox. entrenamiento con la función sin ruido (prueba 2).

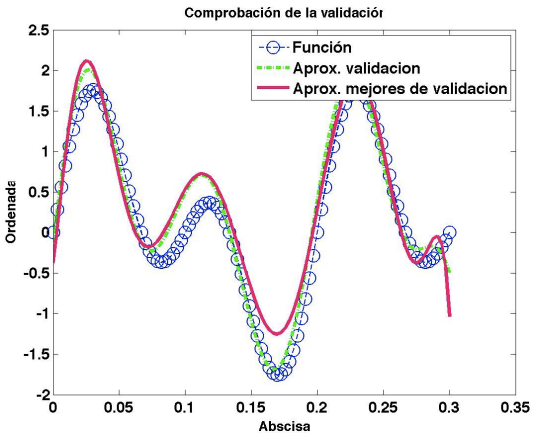


Figura3.62: aprox. validación (prueba 2).

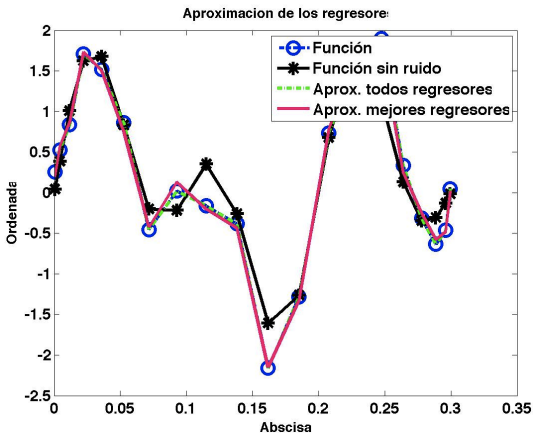


Figura3.63: aprox. entrenamiento con la función sin ruido (prueba 3).

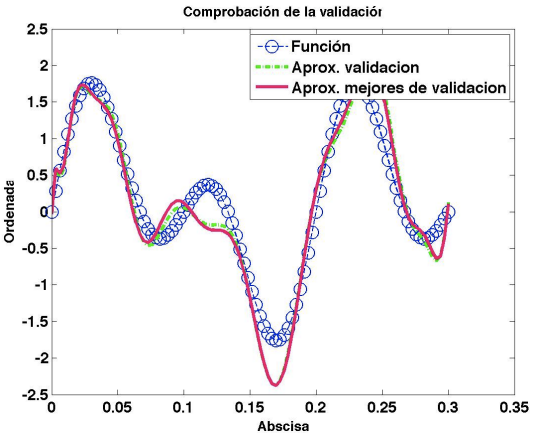


Figura3.64: aprox. validación (prueba 3).

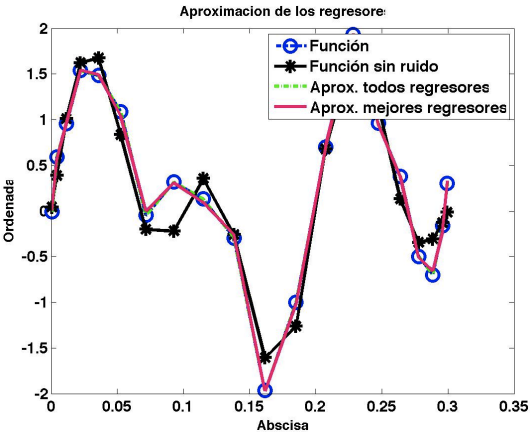


Figura3.65: aprox. entrenamiento con la función sin ruido (prueba 4)**.**

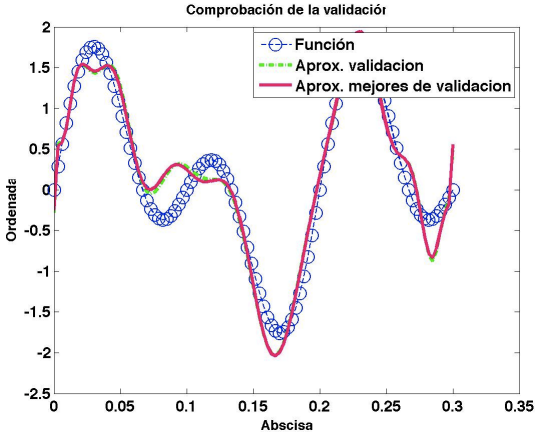


Figura3.66: aprox. validación (prueba 4)**.**

El error de aproximación de validación es, lógicamente, mucho peor cuando existe ruido en los puntos de entrenamiento. Como se observa en las gráficas de aproximación de los puntos de entrenamiento de la red neuronal "aprende" de unos datos que no son los de la función, ya que tienen ruido. Por lo que la red no generaliza bien, la salida de los datos de validación difiere mucho de la función deseada. Cuando mayor sea el ruido (menor SNR), peor aproxima la red neuronal.

4. Redes neuronales de Chebychev unidimensionales en puntos arbitrarios.

4.1 Polinomios de Chebychev en puntos arbitrarios.

En el artículo [32] se muestran una serie de polinomios de Chebychev ($\varphi(x)$) que son ortogonales en un conjunto finito de puntos arbitrarios.

Esto va a permitir, en principio, aplicar el algoritmo OLS de forma óptima en cualquier conjunto de puntos, superando las limitaciones de los polinomios de Chebychev de primera especie. A continuación vamos a mostrar como se generan estos nuevos polinomios.

Tenemos N puntos:

$x_i \quad i = 1,...,N \quad (no\ tienen\ por\ qué\ estar\ equiespaciados, son\ puntos)$

$y_i = 1,...,N \quad (situados\ en\ la\ recta\ "x"\ de\ forma\ arbitraria)$

Queremos encontrar N polinomios (de orden 0 hasta N-1) que sean ortogonales en estos puntos.

Por definición partimos de las siguientes sumas:

$$C_{oq} = \sum_{i=1}^{N} x_i^q \tag{4.1}$$

$$S_{qj} = \sum_{i=1}^{N} x_i^q y_i^j \tag{4.2}$$

Así:

$$C_{00} = \sum_{i=1}^{N} 1 = N; \tag{4.3}$$

Este término se utiliza para calcular los ceros de los polinomios

$$S_{01} = \sum_{i=1}^{N} 1 y_i = \sum_{i=1}^{N} y_i \tag{4.4}$$

Tenemos N parejas de datos de entrenamiento $(x_1, y_1),(x_2, y_2),...,(x_N, y_N)$

Donde:

$$C_{1q} = C_{0,q+1} \quad para\ q \geq 1$$
$$con \quad b_1 = \frac{C_{01}}{C_{00}}\ ,\ C_{01} = \sum_{i=1}^{N} x_i^1 = \sum_{i=1}^{N} x_i \tag{4.5}$$

$$C_{kq} = C_{k-1,q+1} - b_k C_{k-1,q} - a_k C_{k-2,q} \quad para\ q \geq k \geq 2$$
$$con \quad a_k = \frac{C_{k-1,k-1}}{C_{k-2,k-2}} \ , \ b_k = \frac{C_{k-1,k}}{C_{k-1,k-1}} - \frac{C_{k-2,k-1}}{C_{k-2,k-2}} \tag{4.6}$$

Los dos primeros polinomios son:

$$\varphi_0(x) = 1 \tag{4.7}$$

$$\varphi_1(x) = x - b_1 \tag{4.8}$$

El resto de los polinomios se calcula con la fórmula recurrente para $q \geq 2$:

$$\varphi_q(x) = (x - b_q)\varphi_{q-1}(x) - a_q \varphi_{q-2}(x) \tag{4.9}$$

Para cerciorarnos que estos polinomios son ortogonales, vamos a calcular los cinco primeros, así podremos observar este hecho:

$$q = 0 \Rightarrow \varphi_0(x) = 1 \tag{4.10}$$

$$q = 1 \Rightarrow \varphi_1(x) = x - b_1 = x - \frac{C_{01}}{C_{00}} = x - \frac{\sum_{i=1}^{N} x_i}{\sum_{i=1}^{N} 1} = x - \frac{\sum_{i=1}^{N} x_i}{N} \tag{4.11}$$

$$q = 2 \Rightarrow \varphi_2(x) = (x - b_2)\varphi_1(x) - a_2 \varphi_0(x) \tag{4.12}$$

Tenemos que calcular:

$$b_2 = \frac{C_{1,2}}{C_{1,1}} - \frac{C_{0,1}}{C_{0,0}} \tag{4.13}$$

$$a_2 = \frac{C_{1,1}}{C_{0,0}} \tag{4.14}$$

$$C_{1,2} = C_{1,q}\Big|_{q=2} = C_{0,3} - b_1 C_{0,2} = \sum_{i=1}^{N} {x_i}^3 - b_1 \sum_{i=1}^{N} {x_i}^2 \tag{4.15}$$

$$C_{1,1} = C_{1,q}\Big|_{q=1} = C_{0,2} - b_1 C_{0,1} = \sum_{i=1}^{N} {x_i}^2 - b_1 \sum_{i=1}^{N} x_i \tag{4.16}$$

$$q = 3 \Rightarrow \varphi_3(x) = (x - b_3)\varphi_2(x) - a_3 \varphi_1(x) \tag{4.17}$$

Tenemos que calcular:

$$b_3 = \frac{C_{2,3}}{C_{2,2}} - \frac{C_{1,2}}{C_{1,1}} \tag{4.18}$$

$$a_3 = \frac{C_{2,2}}{C_{1,1}} \tag{4.19}$$

$$C_{2,3} = C_{k,q}\Big|_{k=2,q=3} = C_{1,4} - b_2 C_{1,3} - a_2 C_{0,3} \tag{4.20}$$

$$C_{2,2} = C_{k,q}\Big|_{k=2,q=2} = C_{1,3} - b_2 C_{1,2} - a_2 C_{0,2} \tag{4.21}$$

$$q = 4 \Rightarrow \varphi_4(x) = (x - b_4)\varphi_3(x) - a_4\varphi_2(x) \tag{4.22}$$

Tenemos que calcular:

$$b_4 = \frac{C_{3,4}}{C_{3,3}} - \frac{C_{2,3}}{C_{2,2}} \tag{4.23}$$

$$a_4 = \frac{C_{3,3}}{C_{2,2}} \tag{4.24}$$

$$C_{3,4} = C_{k,q}\Big|_{\substack{k=3,q=4 \\ q \geq k \geq 2}} = C_{2,5} - b_3 C_{2,4} - a_3 C_{1,4} \tag{4.25}$$

$$C_{3,3} = C_{k,q}\Big|_{\substack{k=3,q=3 \\ q \geq k \geq 2}} = C_{2,4} - b_3 C_{2,3} - a_2 C_{1,3} \tag{4.26}$$

Los términos C_{kq} se pueden almacenar en una matriz que facilita su cálculo:

$$C = \begin{bmatrix} C_{01} & C_{02} & C_{03} & C_{04} & C_{05} & C_{06} & C_{07} & \cdots \\ C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} & C_{17} & \cdots \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} & C_{27} & \cdots \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} & C_{37} & \cdots \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} & C_{47} & \cdots \end{bmatrix} \tag{4.27}$$

Hemos mostrado la generación sólo de cuatro polinomios, no obstante si tuviéramos que calcular más polinomios, simplemente aplicaríamos la formula de la ecuación recursiva (4.9).

4.2 Estudio de las redes neuronales de Chebychev en puntos arbitrarios.

Como los Polinomios de Chebychev pueden utilizar puntos regulares y arbitrarios, en cada uno de los siguientes apartados mostraremos los dos estudios.

4.2.1 Estudio de la ortogonalidad de los regresores (matriz H). En puntos regulares y arbitrarios.

Vamos a mostrar los problemas que existen para generar una matriz de regresores sea ortogonal, es decir, la matriz H no siempre será diagonal.

Sobre el rango de la matriz de regresores. Esta matriz, es una matriz cuadrada, el número de filas es igual al número de columnas que es igual al número de puntos de entrenamiento (n). Los regresores deberían ser ortogonales; si son ortogonales entonces son linealmente independientes y en este caso el rango de la matriz será igual a n y el determinante es distinto de cero. En cambio, si los regresores no son linealmente independientes, entonces el rango es menor que n y el determinante es cero. Si los regresores no son linealmente independientes, seguro que no son ortogonales. Podría darse el caso además de que fueran linealmente independientes y no fueran ortogonales, por eso medimos el ratio diagonal de la matriz H y vemos cómo es esa matriz. De todos modos, en general en cuanto el rango no sea n, el determinante es cero o muy cercano a 0 y los regresores no son ortogonales y además no son linealmente independientes. A veces el determinante es casi cero pero el rango es n. Pero esto es debido a que Matlab utiliza mucha precisión y utiliza métodos muy avanzados para calcular el rango. En estos casos estamos cerca de la zona peligrosa en la que los regresores ya no son ortogonales.

Lo que ocurre con nuestra matiz de regresores es que aparecen columnas en las que todos sus elementos valen cero. Eso hace que el rango se reduzca y el determinante valga 0.

- Tabla y gráficas del estudio de la función 3.10 en puntos regulares.

En la tabla 4.1 se muestran los datos para medir la ortogonalidad de los regresores en la función 3.10. El número de puntos de entrenamiento n, será 4 y 20, y el intervalo (rango de trabajo) de esta función es [-3,3].

PRUEBA	Inicial	Final	n
1	1	1.8	4
2	1	3	4
3	1	5	4
4	0	0.5	20
5	1	1.8	20
6	1	3	20

Tabla 4.1: Datos de las pruebas para medir la ortogonalidad de los regresores de la función 3.10 en puntos regulares.

En la siguiente tabla mostramos los resultados que miden si la matriz H es diagonal en las pruebas realizadas.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	1.094	4.294e-15	2.548e+14
2	1.792	3.775e-14	4.748e+13
3	8.911	3.938-14	2.263e+14
4	1.024	3.269e-18	3.131e+17
5	1.062	3.445e-07	3.082e+06
6	1.752e+03	72.882	24.037

Tabla 4.2: Parámetro de medición de la ortogonalidad (ratio diagonal) de la función 3.10 en puntos regulares.

Gráficas:

En todas las gráficas que se mostrarán en el apartado 4.2.1., las abscisas representan las columnas y las ordenadas las filas.

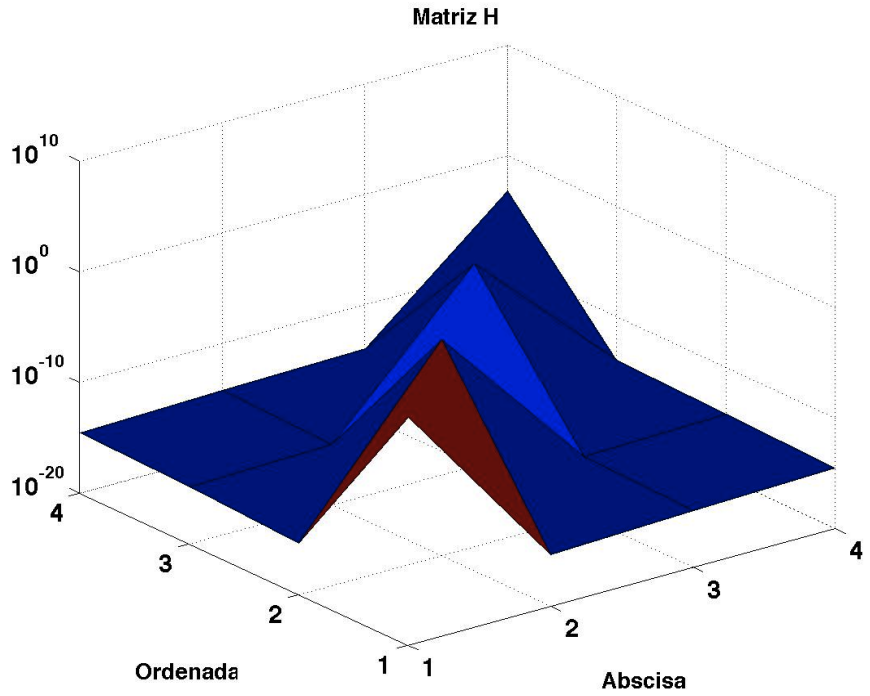


Figura 4.1: Matriz H con pocos puntos de entrenamiento y un rango pequeño (prueba 1).

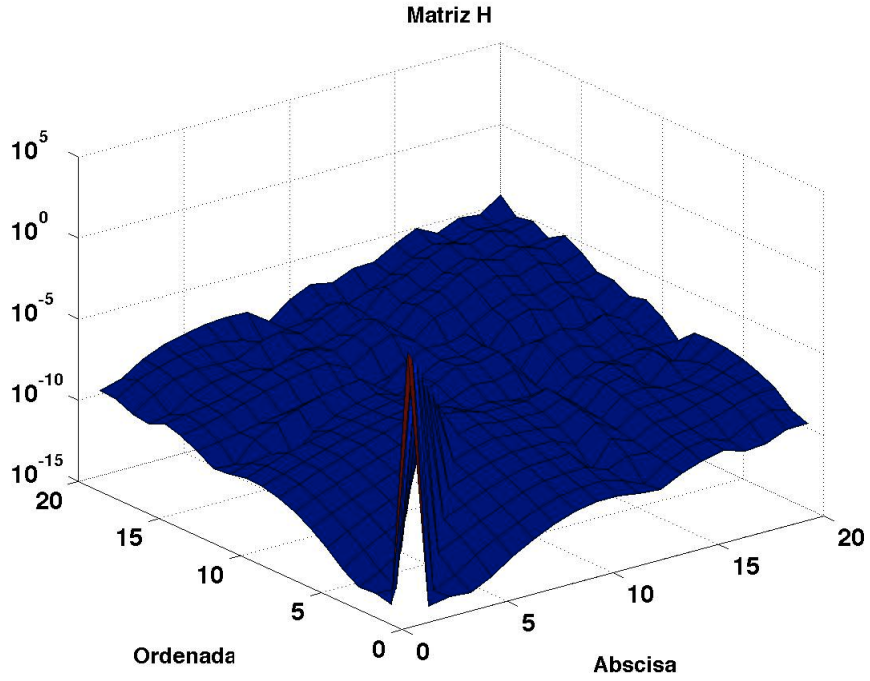


Figura 4.2: Matriz H con muchos puntos de entrenamiento y un rango medio (prueba 5).

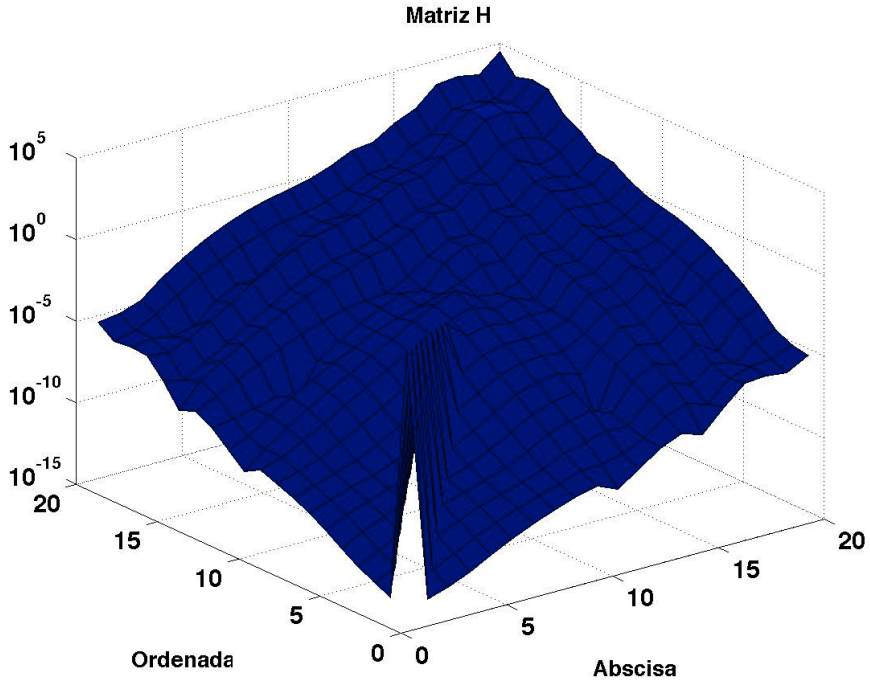


Figura 4.3: Matriz H con muchos puntos de entrenamiento y un rango grande (prueba 6).

- Tabla y gráficas del estudio de la función 3.11 en puntos regulares.

En la tabla 4.4 se muestran los datos para las pruebas para medir la ortogonalidad de los regresores en la función 3.11. El número de puntos de entrenamiento n, será 4 y 20, y el intervalo (rango de trabajo) de esta función es [0,0.3].

PRUEBA	Inicial	Final	n
1	1	1.8	4
2	1	3	4
3	1	5	4
4	0	0.5	20
5	1	1.8	20
6	1	3	20

Tabla 4.3: Datos de las pruebas para medir la ortogonalidad de los regresores en la función 3.11 en puntos regulares.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	1.094	4.294e-15	2.548e+14
2	1.793	3.775e-14	4.748e+13
3	8.911	3.938-14	2.263e+14
4	1.024	3.269e-18	3.131e+17
5	1.062	3.445e-07	3.082e+06
6	1.752e+03	72.882	24.037

Tabla 4.4. Parámetro de medición de la ortogonalidad (ratio diagonal) en la función 3.11 en puntos regulares.

Gráficas:

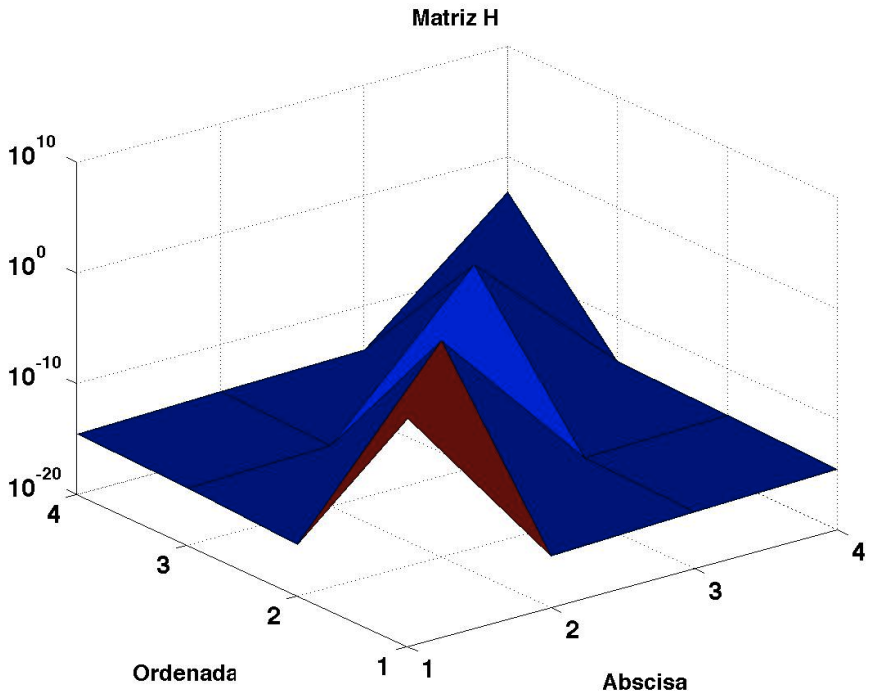


Figura 4.4: Matriz H con pocos puntos de entrenamiento y un rango mediano (prueba 1).

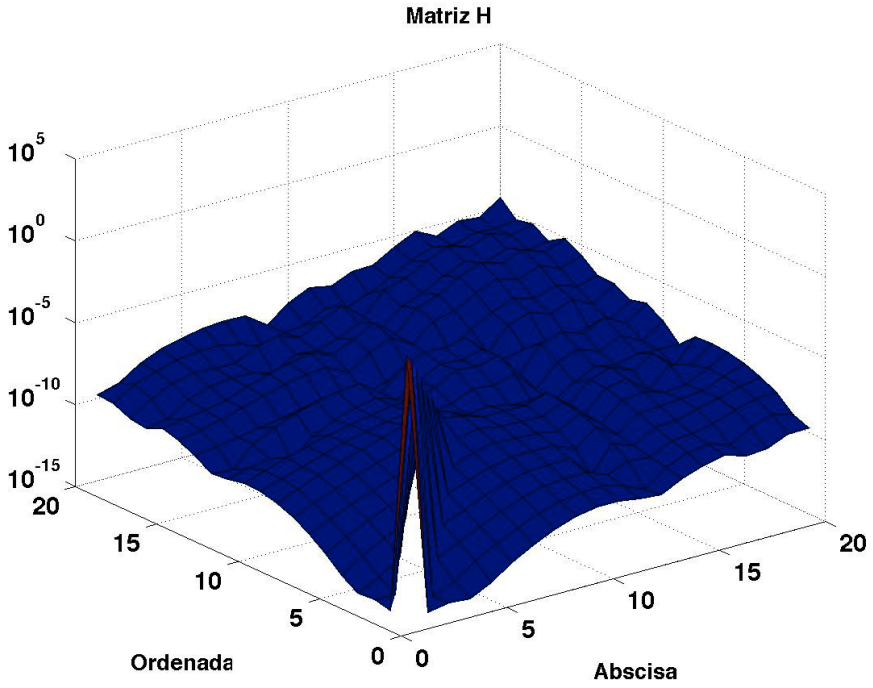


Figura 4.5: Matriz H con muchos puntos de entrenamiento y un rango mediano (prueba 5).

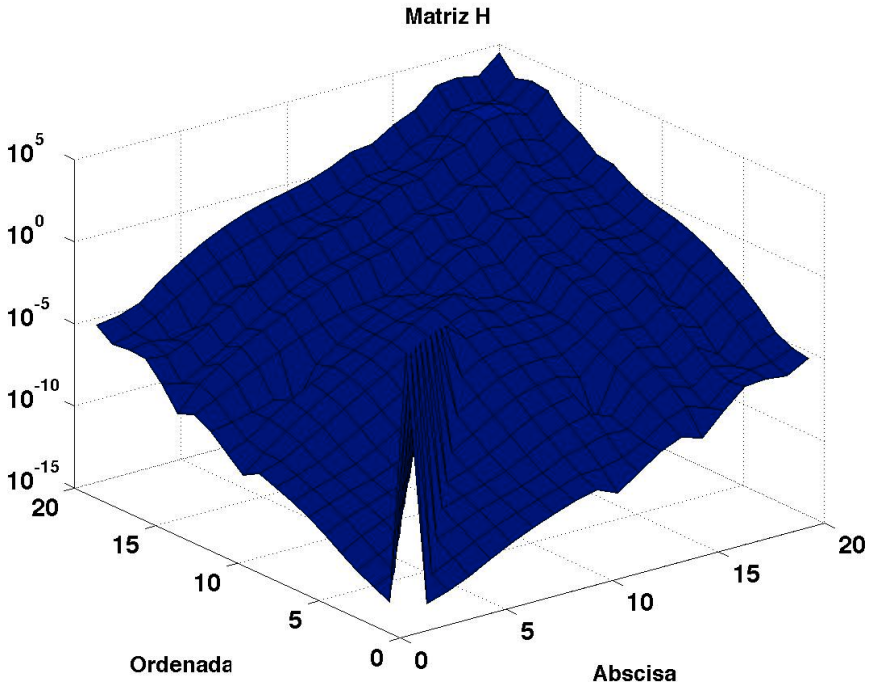


Figura 4.6: Matriz H con muchos puntos de entrenamiento y un rango grande (prueba 6).

A continuación estudiaremos lo que ocurre con la ortogonalidad cuando se generan puntos arbitrarios.

- Tabla y gráficas del estudio de la función 3.10 en puntos arbitrarios.

PRUEBA	Inicial	Final	n
1	1	1.5	4
2	1	3	4
3	1	5	4
4	0	0.3	20
5	1	1.5	20
6	1	3	20

Tabla 4.5: Datos de las pruebas para medir la ortogonalidad de los regresores en la función 3.10 en puntos arbitrarios.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	1.1107	1.5054e-16	7.3781e+15
2	1.1733	1.1376e-14	1.0314e+14
3	3.3274	1.6866-13	1.9728e+13
4	1.0037	6.7660e-19	1.4834e+18
5	1.0184	1.9726e-10	5.1625e+09
6	317.1897	8.5316	37.1781

Tabla 4.6. Parámetro de medición de la ortogonalidad (ratio diagonal) en la función 3.10 en puntos arbitrarios.

Gráficas:

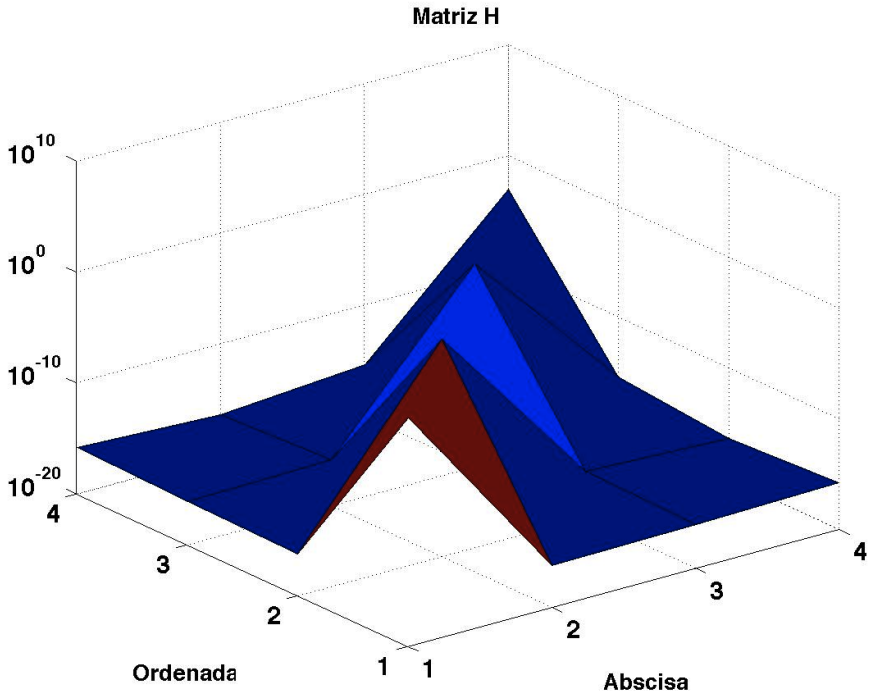


Figura 4.7: Matriz H con pocos puntos de entrenamiento y un rango pequeño (prueba 1).

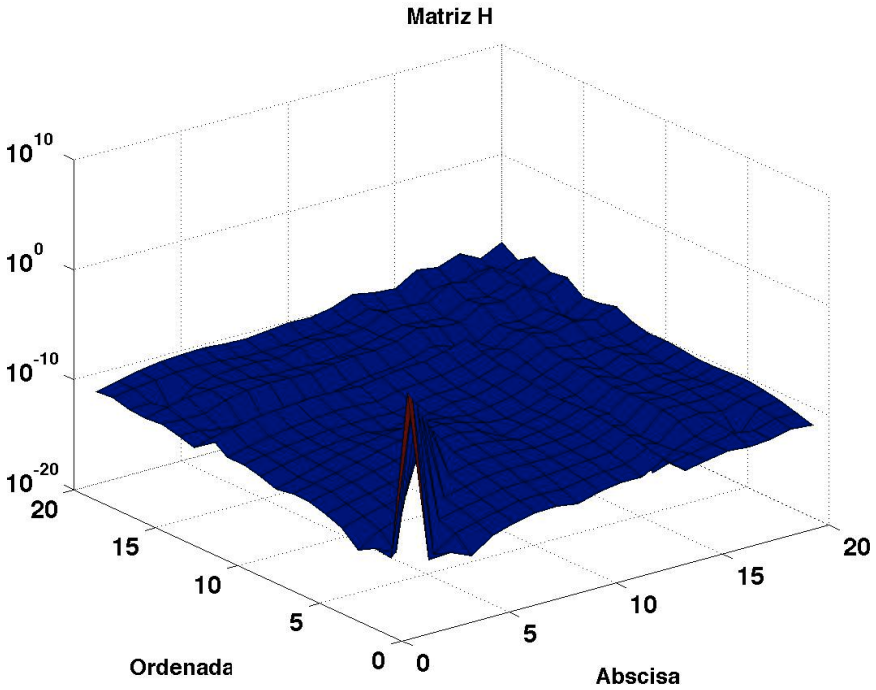


Figura 4.8: Matriz H con muchos puntos de entrenamiento y un rango pequeño (prueba 5).

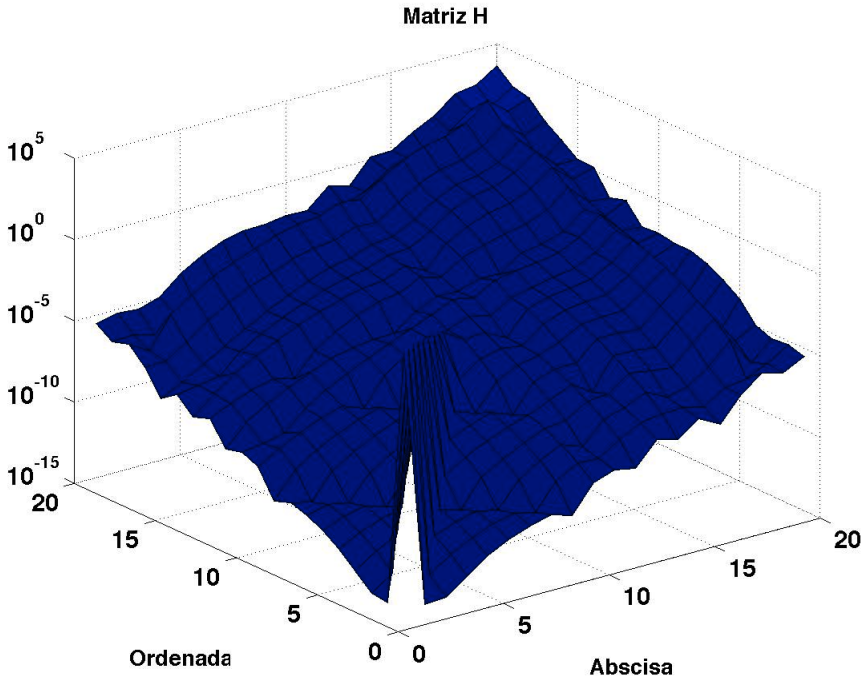


Figura 4.9: Matriz H con muchos puntos de entrenamiento y un rango grande (prueba 6).

- Tabla y gráficas del estudio de 3.11 en puntos arbitrarios.

PRUEBA	Inicial	Final	n
1	1	1.5	4
2	1	3	4
3	1	5	4
4	1	1.5	20
5	1	3	20
6	1	5	20

Tabla 4.7: Datos de las pruebas para medir la ortogonalidad de los regresores en la función 3.11 en puntos arbitrarios.

PRUEBA	Media diagonal	Media no diagonal	Ratio diagonal
1	1.0140	3.7397e-15	2.7115e+14
2	1.1951	2.9910e-15	3.9958e+14
3	4.0711	4.9840-14	8.1683e+13
4	1.0284	1.7043e-10	6.0340e+09
5	63.4577	3.4510	18.3884
6	6.2791e+11	2.8425e+09	220.9020

Tabla 4.8. Parámetro de medición de la ortogonalidad (ratio diagonal) en la función 3.11 en puntos arbitrarios.

Gráficas:

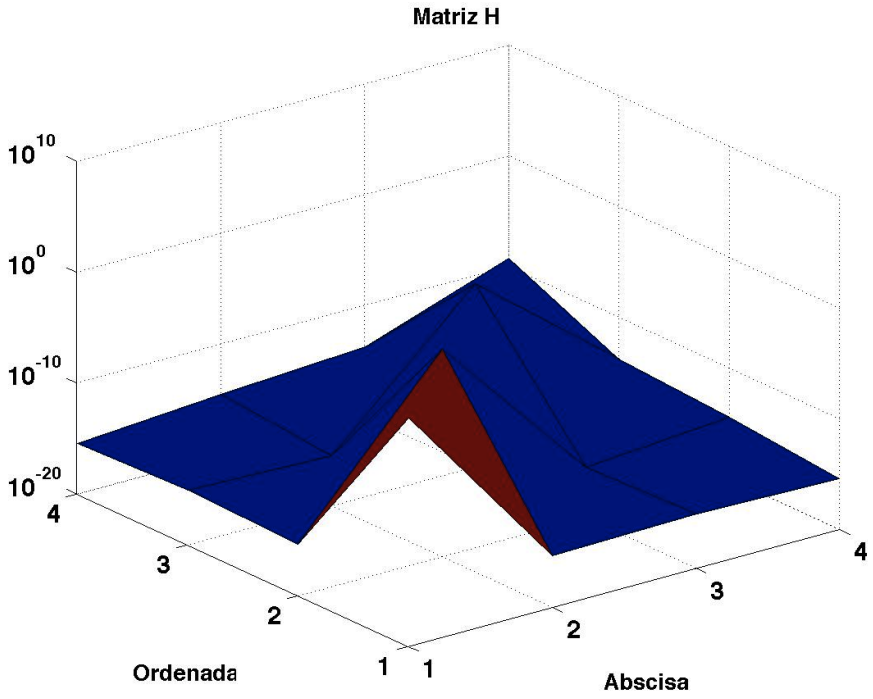


Figura 4.10: Matriz H con pocos puntos de entrenamiento y un rango pequeño (prueba 1).

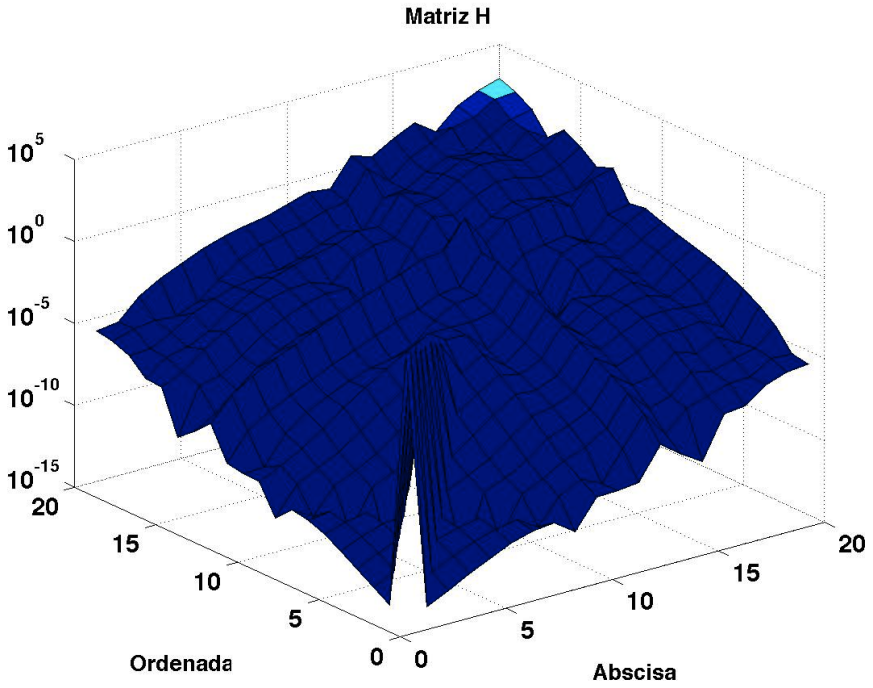


Figura 4.11: Matriz H con muchos puntos de entrenamiento y un rango grande (prueba 5).

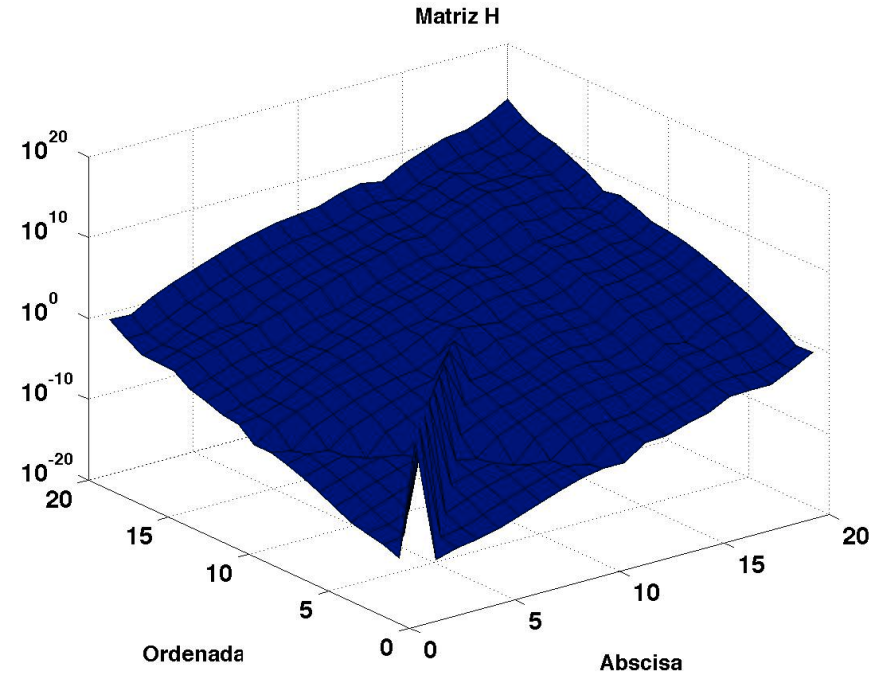


Figura 4.12: Matriz H con muchos puntos de entrenamiento y un rango grande (prueba 6).

Viendo los resultados tanto para puntos regulares como arbitrarios, se llegan a las siguientes conclusiones:

Cuando hay pocos puntos de entrenamiento el rango es siempre n y la matriz H es diagonal (regresores ortogonales).

Cuando hay muchos puntos de entrenamiento el método no funciona bien, los regresores dejan de ser ortogonales. El rango de la matriz de regresores se reduce, ya no vale n, vale un valor menor que n. Además el determinante vale cerca de cero. La matriz H deja de ser diagonal; en ocasiones para muchos puntos n=20 obtenemos un ratio diagonal elevado, como si la matriz H fuese diagonal pero si observamos la figura de H lo que ocurre es que hay unos valores en la diagonal demasiado grandes y valores fuera de la diagonal demasiados pequeños, pero también muy grandes. Eso hace que a la vez obtengamos un valor ratio diagonal elevado y que la matriz H no sea diagonal.

Además cuando hay muchos puntos de entrenamiento da igual que aumentemos el rango. Siempre hay problemas en la generación de la matriz de regresores y los regresores siempre son no ortogonales.

Con pocos puntos no hay problema excepto si reducimos el intervalo en cuyo caso aparecen columnas (regresores) cuyos valores valen cero, y si utilizamos muchos puntos de entrenamiento entonces el método no funciona. Con puntos arbitrarios ocurre lo mismo, lo que pasa es que con pocos puntos de entrenamiento si algunos están muy cerca unos de otros, el determinante puede llegar a valer cerca de cero.

4.2.2. Estudio del número de puntos de entrenamiento necesarios. En puntos regulares y arbitrarios.

Vamos a ver que cuando aumentamos el número de puntos de entrenamiento la aproximación mejora. Mejora el error de aproximación en los datos de entrenamiento y de validación. Sin embargo, la matriz de regresores no siempre será ortogonal.

- Tabla y gráficas del estudio de la función 3.9 en puntos regulares.

En la tabla 4.9 se muestran los datos para las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.9. El número de puntos de entrenamiento n, varía en 5, 7 y 10, y el intervalo (rango de trabajo) de esta función es [-2,2].

PRUEBA	Inicial	Final	n	n val
1	-2	2	**5**	100
2	-2	2	**7**	100
3	-2	2	**10**	100

Tabla 4.9: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.9 en puntos regulares.

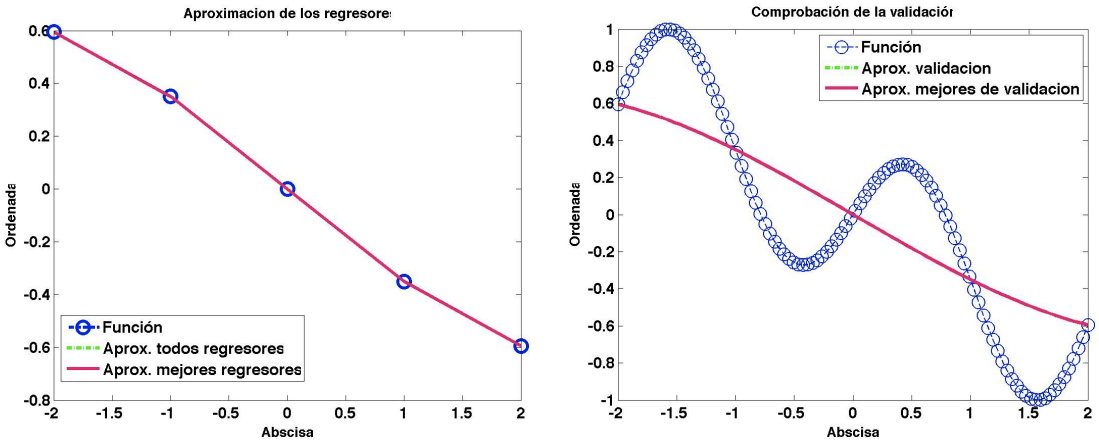
Para medir de forma cuantitativa la precisión de la red neuronal hemos utilizado un parámetro de medición del error que se denomina FVU, éste está expresando en la ecuación 3.12.

PRUEBA	Error FVU entrenamiento	Error FVU validación
1	0e+00	3.071e-01
2	2.272e-32	5.837e-02
3	8.524e-32	1.946e-04

Tabla 4.10: Parámetros de error de entrenamiento y validación de la función 3.9 en puntos regulares.

Gráficas:

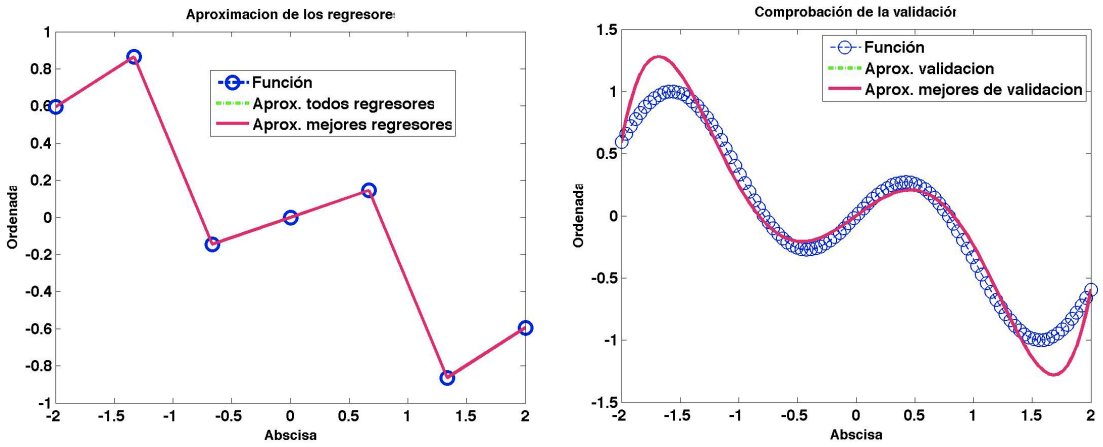
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 4.13: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

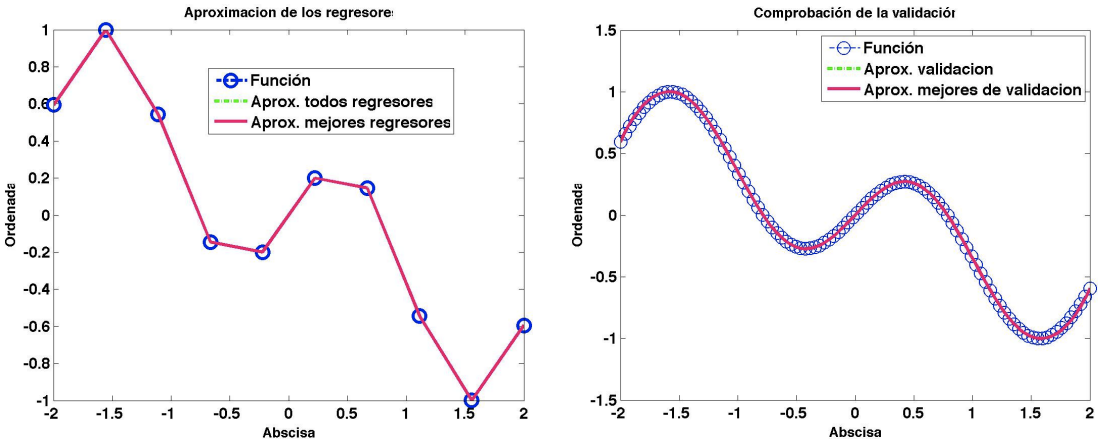
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 4.14: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

* Diez puntos de entrenamiento.



(a) **(b)**

Figura 4.15: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

Como observamos en las gráficas anteriores, con 5 puntos la función no es bien muestreada y la salida de la red neuronal se parece a recta, con 7 puntos la función es mejor muestreada y con 10 el muestreo es suficiente para conseguir una buena aproximación en entrenamiento y validación. De nuevo, vuelve a ocurrir lo mismo que con Chebychev no arbitrario.

- Tabla y gráficas del estudio de la función 3.10 en puntos regulares.

PRUEBA	Inicial	Final	n	n val
1	-3	3	**5**	100
2	-3	3	**7**	100
3	-3	3	**15**	100

Tabla 4.11: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.10 en puntos regulares.

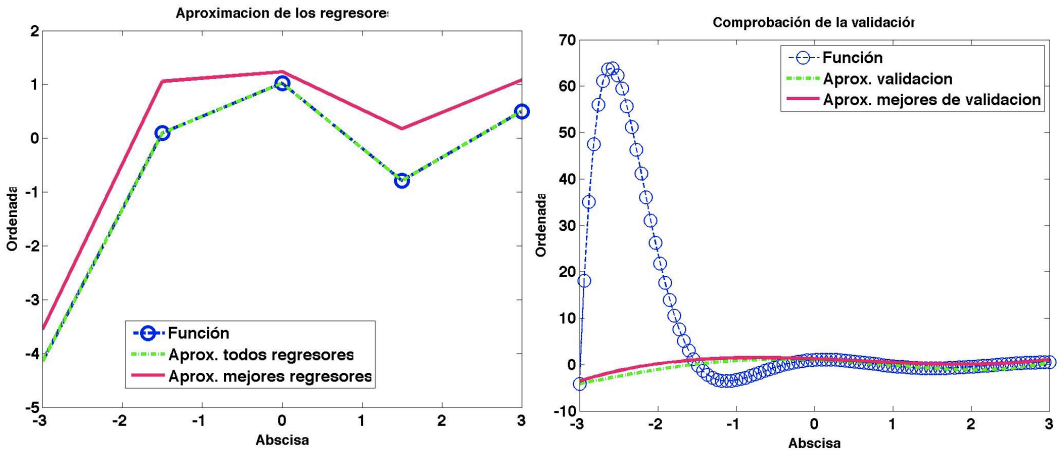
PRUEBA	Error FVU entrenamiento	Error FVU validación
1	2.014e-31	1.311e+00
2	5.518e-32	2.269e-01
3	1.166e-31	3.417e-07

Tabla 4.12: Parámetros de error de entrenamiento y validación de la función 3.10 en puntos regulares.

En la tabla 4.12, el error es muy pequeño y se reduce en gran medida con el aumento de los puntos de entrenamiento.

Gráficas:

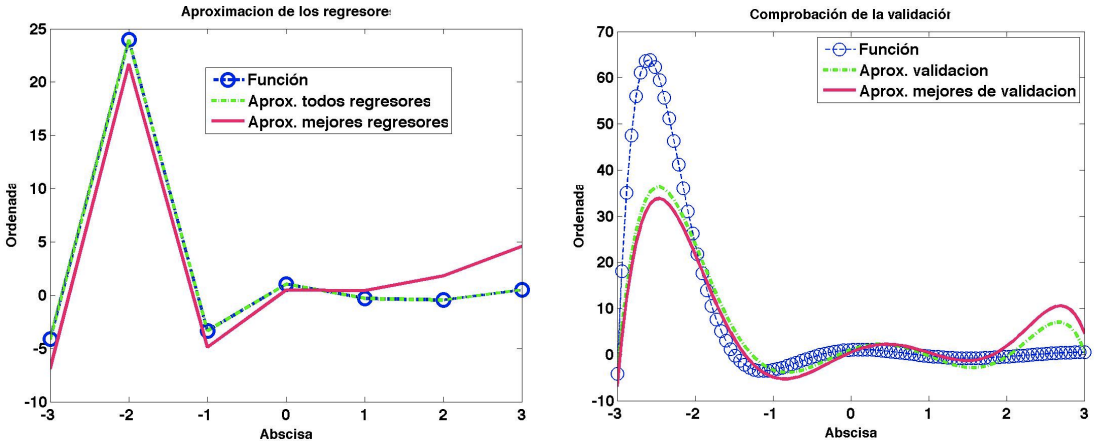
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 4.16: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

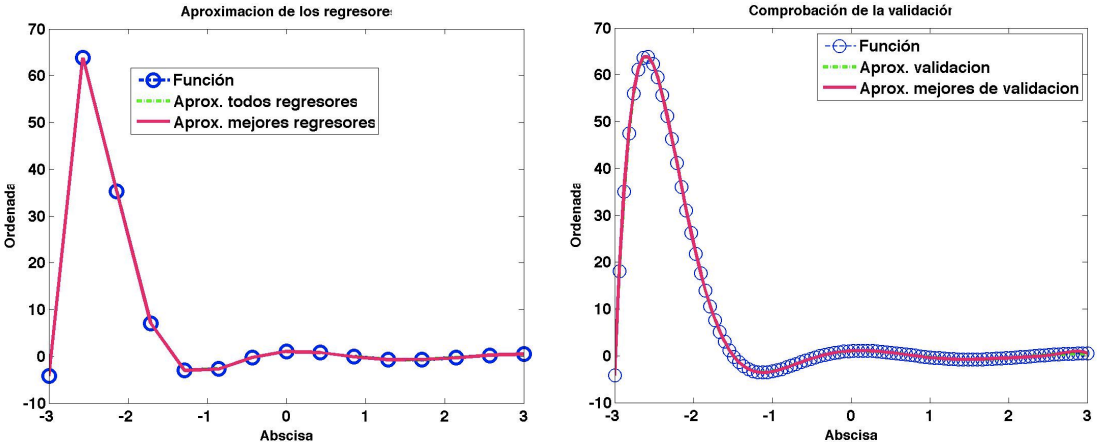
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 4.17: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

* Quince puntos de entrenamiento.



(a) **(b)**

Figura 4.18: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=15 (prueba 3).

Como se puede observar en la primera gráfica, como solo tenemos 5 puntos de entrenamiento, la función no se muestrea adecuadamente, no tenemos suficiente información de cómo cambia la función, especialmente cerca de -3, donde la función cambia rápidamente. Esta es una función difícil de aproximar. En cambio, cuando tenemos 7 puntos de entrenamiento (añadimos 2 puntos de entrenamiento) la función se representa mucho mejor. La aproximación mejora. Cuando tenemos 15 puntos de

entrenamiento la aproximación tanto en los datos de entrenamiento como sobre todo en los datos de validación es muy precisa.

Como se puede apreciar, volvemos a las mismas conclusiones que en el caso de Chebychev no arbitrario.

- Tabla y gráficas del estudio de la función 3.11 en puntos regulares.

PRUEBA	Inicial	Final	n	n val
1	0	0.3	**5**	100
2	0	0.3	**7**	100
3	0	0.3	**10**	100
4	0	0.3	**15**	100

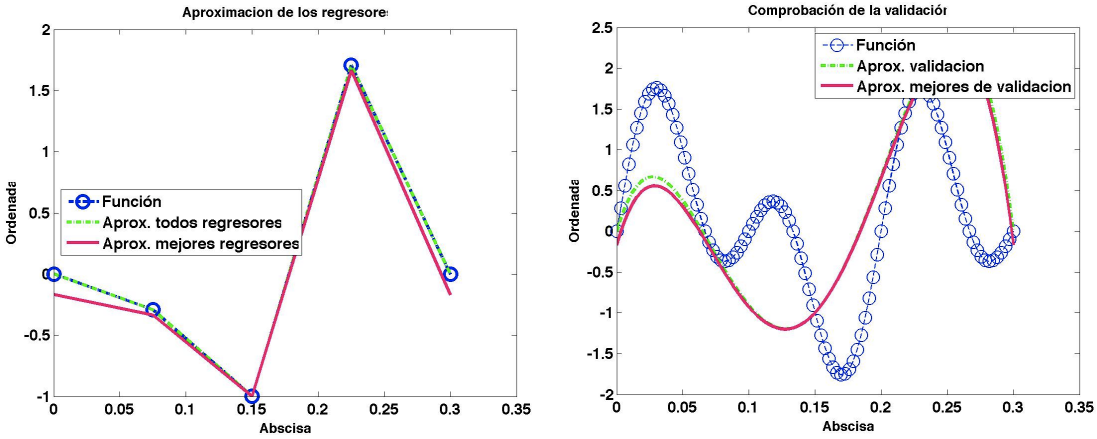
Tabla 4.13: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.11 en puntos regulares.

PRUEBA	Error FVU entrenamiento	Error FVU validación
1	3.416e-33	1.231e+00
2	1.153e-31	5.260e-01
3	8.744e-32	7.370e-01
4	4.220e-31	1.794e-03

Tabla 4.14: Parámetros de error de entrenamiento y validación de la función 3.11 en puntos regulares.

Gráficas:

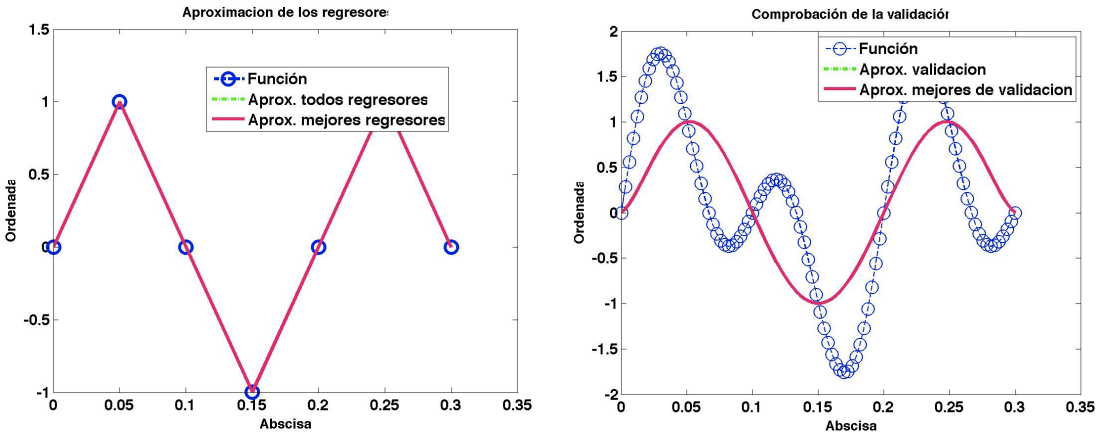
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 4.19: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

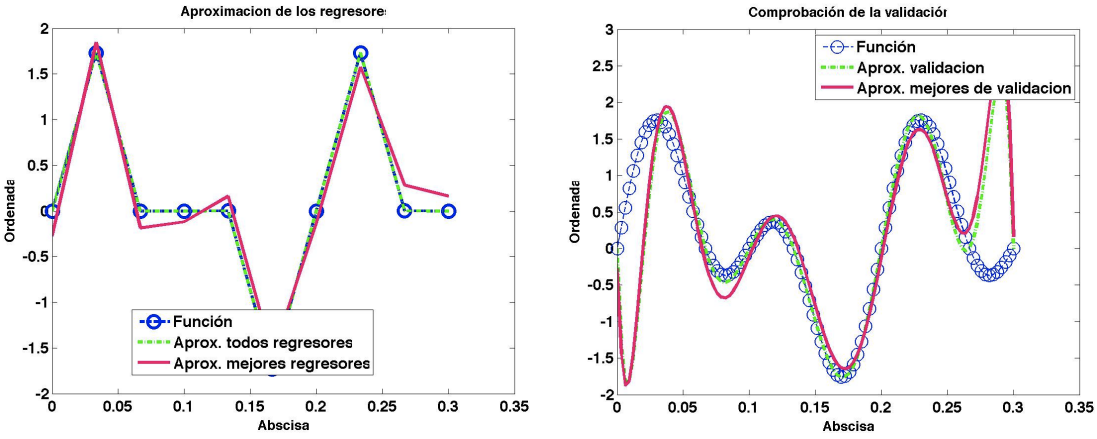
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 4.20: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

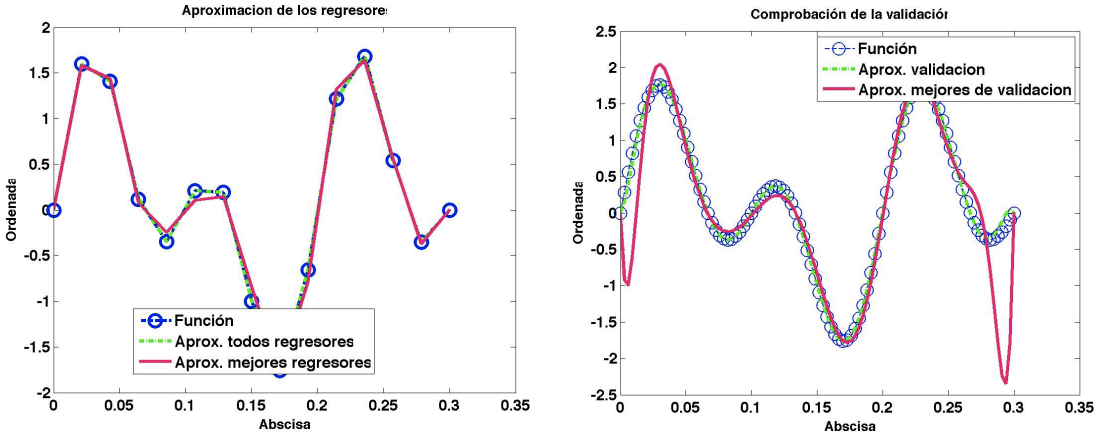
* Diez puntos de entrenamiento.



(a) **(b)**

Figura 4.21: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

* Quince puntos de entrenamiento.



(a) **(b)**

Figura 4.22: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=15 (prueba 4).

En esta función mostramos un caso mas, ya que con 7 puntos de entrenamiento la representación no es todavía adecuada. La función presenta cambios y oscilaciones en el rango [0,0.3] que hacen necesario por lo menos 10 puntos de entrenamiento para una representación mínima de cómo es la función.

De nuevo, volvemos a las mismas conclusiones que en el caso de Chebychev no arbitrario.

Ahora estudiaremos como es la aproximación cuando se generan números aleatorios.

- Tabla y gráficas del estudio de la función 3.9 en puntos aleatorios.

En la tabla 4.15 se muestran los datos para las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.9 en puntos aleatorios. El número de puntos de entrenamiento n, varía en 5, 7 y 10, y el intervalo (rango de trabajo) de esta función es [-2,2].

PRUEBA	Inicial	Final	n	n val
4	-2	2	**5**	100
5	-2	2	**7**	100
6	-2	2	**10**	100

Tabla 4.15: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.9 en puntos aleatorios.

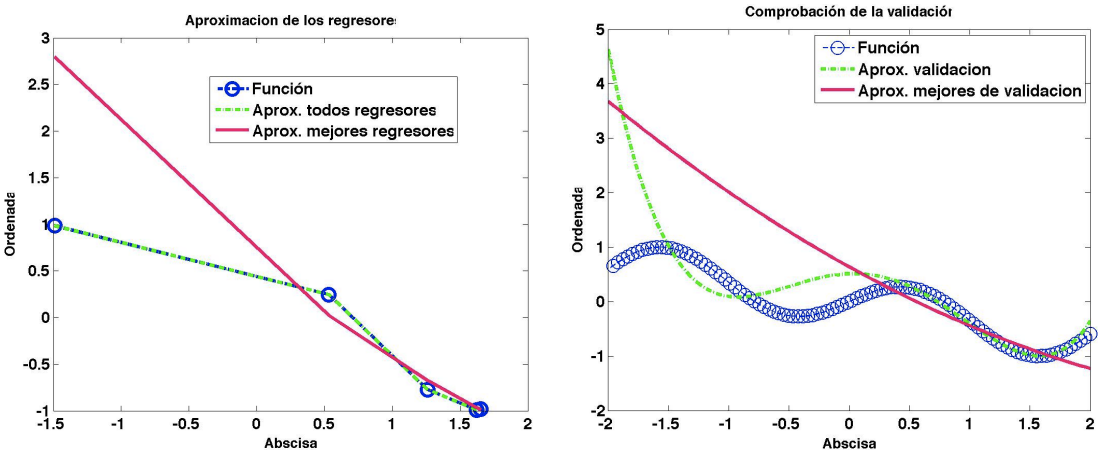
Para medir de forma cuantitativa la precisión de la red neuronal hemos utilizado un parámetro de medición del error que se denomina FVU, éste está expresando en la ecuación 3.12.

PRUEBA	Error FVU entrenamiento	Error FVU validación
4	1.985e-31	1.873e+00
5	2.047e-32	2.913e-01
6	1.556e-31	6.068e-02

Tabla 4.16: Parámetros de error de entrenamiento y validación de la función 3.9 en puntos aleatorios.

Gráficas:

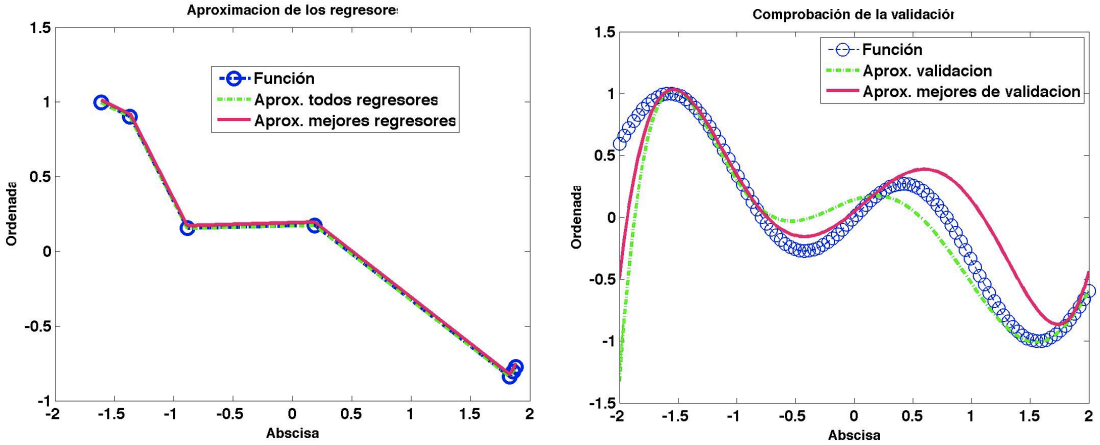
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 4.23: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

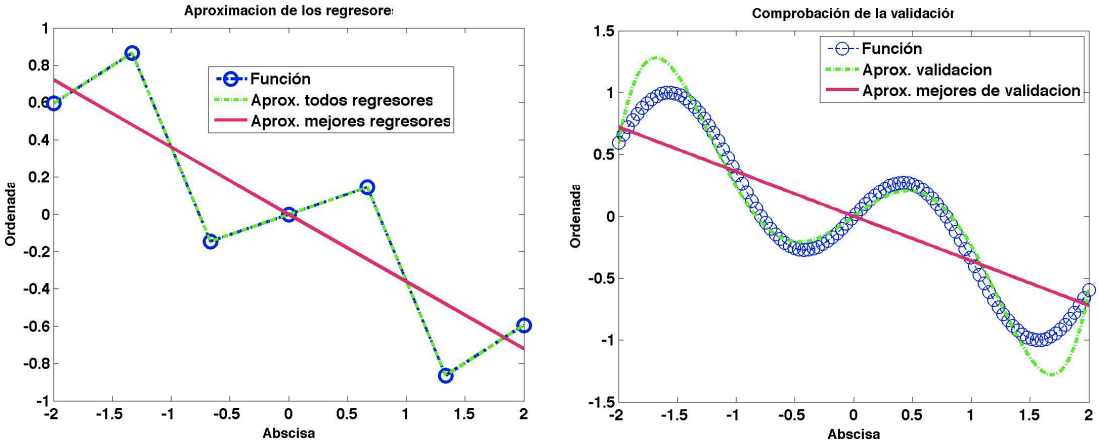
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 4.24: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

* Diez puntos de entrenamiento.



(a) **(b)**

Figura 4.25: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

Como podemos observar, con 5 puntos la función no es bien muestreada y la salida de la red neuronal se parece a recta, con 7 puntos la función es mejor muestreada y con 10 el muestreo permite mejorar la aproximación, pero no es tan buena como con puntos regulares.

- Tabla y gráficas del estudio de la función 3.10 en puntos aleatorios.

PRUEBA	Inicial	Final	n	n val
1	-3	3	**5**	100
2	-3	3	**7**	100
3	-3	3	**15**	100

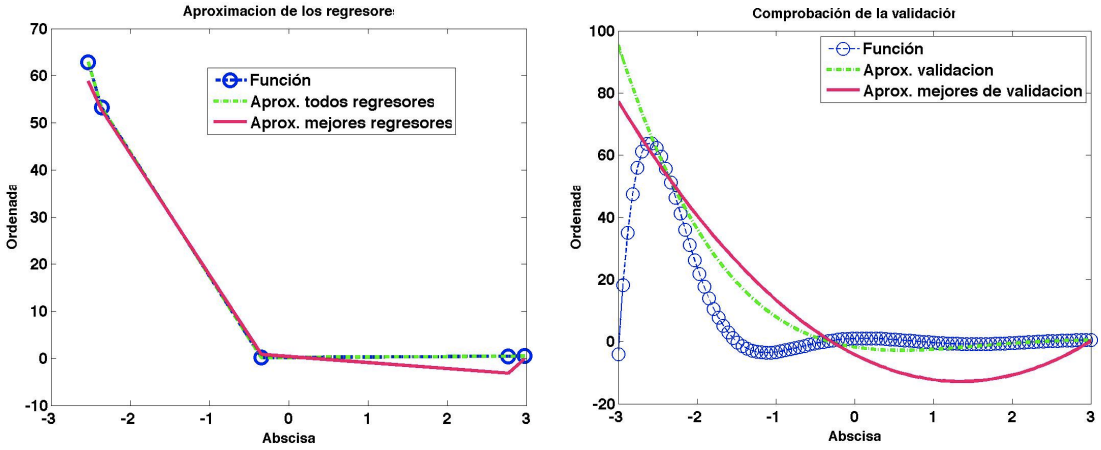
Tabla 4.17: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.10 en puntos aleatorios.

PRUEBA	Error FVU entrenamiento	Error FVU validación
1	3.937e-32	7.432e-01
2	2.969e-32	5.949e-01
3	7.585e-32	1.664e-04

Tabla 4.18: Parámetros de error de entrenamiento y validación de la función 3.10 en puntos aleatorios.

Gráficas:

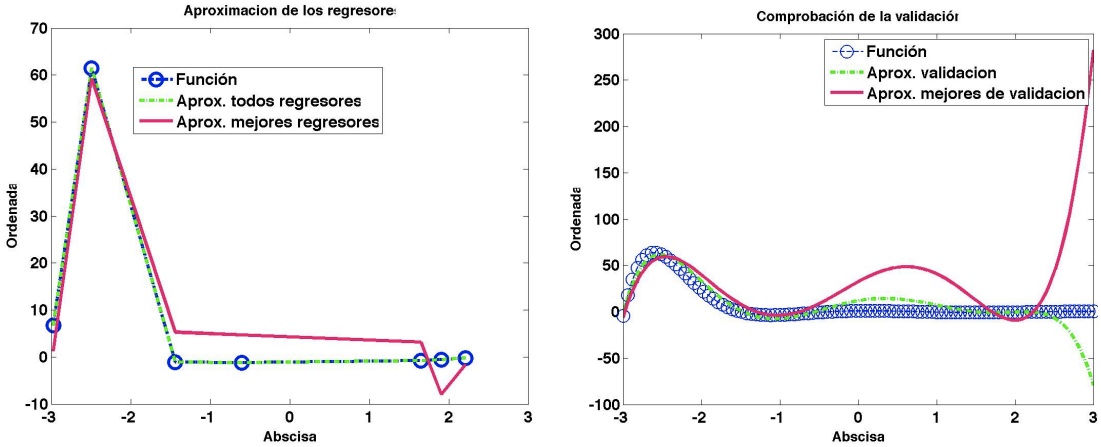
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 4.26: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

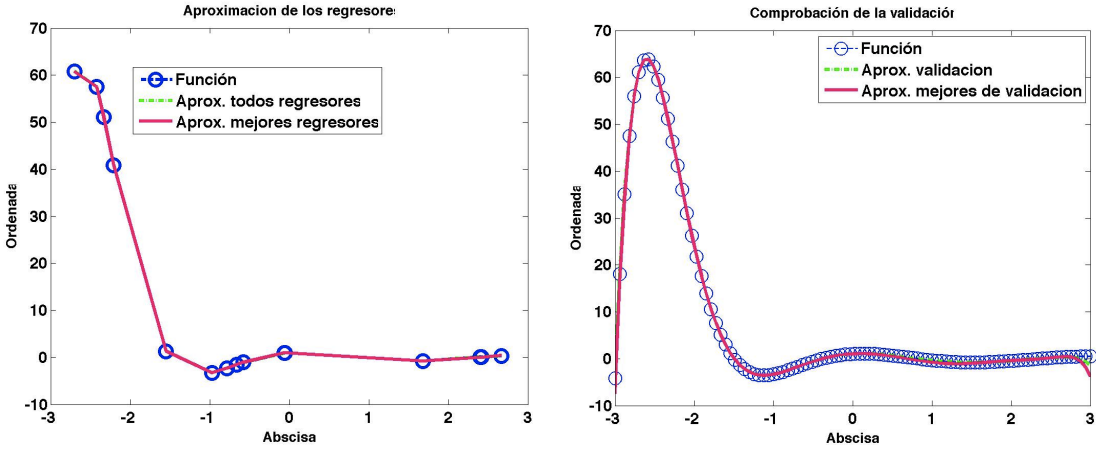
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 4.27: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

* Quince puntos de entrenamiento.



(a) **(b)**

Figura 4.28: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=15 (prueba 3).

Ahora podemos observar que con pocas neuronas (5 y 7 puntos de entrenamiento) la aproximación posee poca precisión ya que al tener puntos aleatorios se distribuyen en unas zonas y en otras no. Por esta razón, hay zonas de la función para las que no existen puntos de entrenamiento y por lo tanto es muy difícil que la red generalice correctamente en esas zonas. Con 15 puntos de entrenamiento y 15 neuronas la aproximación es muy buena.

- Tabla y gráficas del estudio de la función 3.11 en puntos aleatorios.

PRUEBA	Inicial	Final	n	n val
1	0	0.3	**5**	100
2	0	0.3	**7**	100
3	0	0.3	**10**	100
4	0	0.3	**15**	100
5	0	0.3	**25**	100

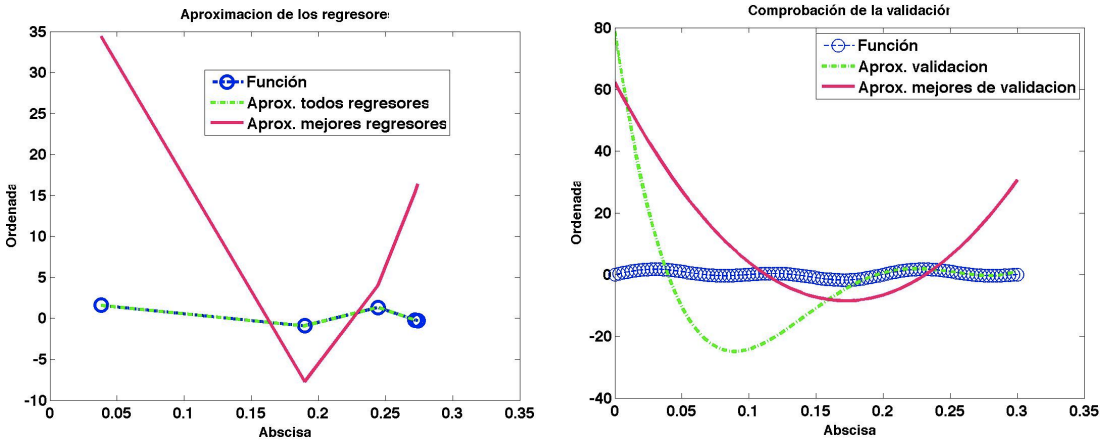
Tabla 4.19: Datos de las pruebas para estudiar el número de puntos de entrenamiento necesarios para la función 3.11 en puntos aleatorios.

PRUEBA	Error FVU entrenamiento	Error FVU validación
1	1.495e-28	3.998e+02
2	4.024e-32	1.313e+00
3	6.390e-29	1.503e+02
4	8.677e-31	8.795e+00
5	45.543e-28	2.636e-09

Tabla 4.20: Parámetros de error de entrenamiento y validación de la función 3.11 en puntos aleatorios.

Gráficas:

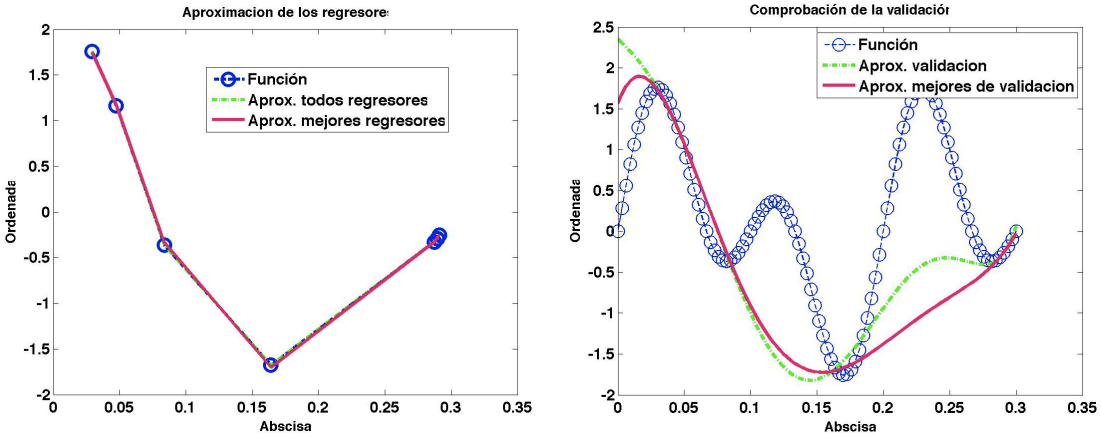
* Cinco puntos de entrenamiento.



(a) **(b)**

Figura 4.29: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=5 (prueba 1).

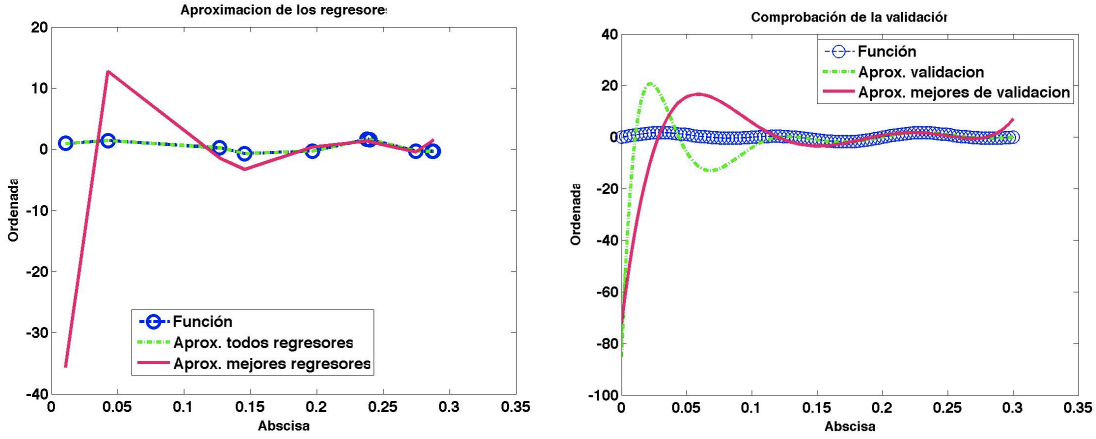
* Siete puntos de entrenamiento.



(a) **(b)**

Figura 4.30: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=7 (prueba 2).

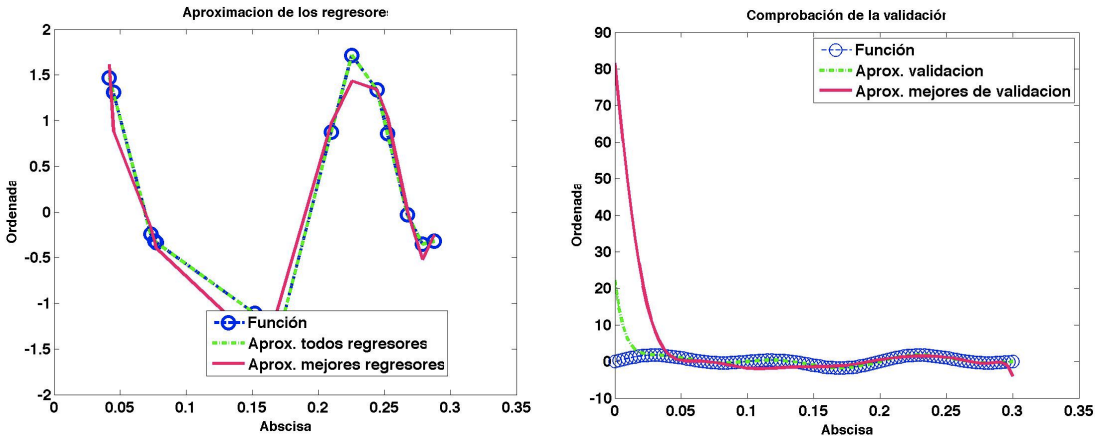
* Diez puntos de entrenamiento.



(a) **(b)**

Figura 4.31: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=10 (prueba 3).

* Quince puntos de entrenamiento.



(a) **(b)**

Figura 4.32: Aproximación en los puntos de entrenamiento (a) y validación (b) cuando n=15 (prueba 4).

Hemos mostrado un caso mas que en los ejemplos anteriores con las funciones 3.9 y 3.10. Ni siquiera con 25 puntos de entrenamiento y 25 neuronas, conseguimos una aproximación adecuada cuando los puntos son arbitrarios.

4.2.3. Estudio del número de regresores necesarios. En puntos regulares y arbitrarios.

En este apartado también estudiaremos las tres funciones por separado y finalizaremos con una única conclusión final, ya que a las tres funciones le sucede lo mismo.

- Tabla y gráficas del estudio de la función 3.9 en puntos regulares.

En la siguiente tabla se muestran los datos que hemos utilizado para estudiar el número de polinomios necesarios para la aproximación de la función 3.9 en puntos regulares.

En esta tabla 4.21, número neuronas mejores, indica el número de neuronas (polinomios) que forman la red neuronal. Si por ejemplo el número de neuronas es 3, se utilizan tres polinomios de entre los polinomios de orden 0 hasta 6 (n=7).

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	-2	2	7	100	**1**
2	-2	2	7	100	**2**
3	-2	2	7	100	**3**
4	-2	2	7	100	**4**

Tabla 4.21 : Datos de la prueba para estudiar el número de regresores necesarios para la función 3.9 en puntos regulares.

Para medir la calidad de la aproximación de las redes neuronales estudiadas, recurrimos de nuevo al estudio cualitativo mediante gráficas, y cuantitativo mediante el error FVU. Además, como se observa en la tabla 4.22, también medimos la calidad de aproximación mediante un parámetro que denominamos porcentaje de energía; éste parámetro mide el cociente de la energía de la señal deseada, expresada según la ecuación (2.16) y la energía de los valores de salida de la red neuronal. Hay que tener en cuenta que el valor máximo de porcentaje de energía es uno.

PRUEBA	Error FVU entrenamiento (mejores regresores)	Error FVU validación (mejores regresores)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	2.783e-01	2.659e-01	1	0.722
2	4.610e-03	6.417e-02	1	0.995
3	1.739e-31	5.837e-02	1	1
4	1.739e-31	5.837e-02	1	1

Tabla 4.22 : Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.9 en puntos regulares.

Gráficas:

Vamos a mostrar únicamente las gráficas de aproximación de validación de las pruebas 1, 2, 3 y 4; y una sola gráfica en la que se muestre la contribución de validación. Ésta será la de la prueba 1.

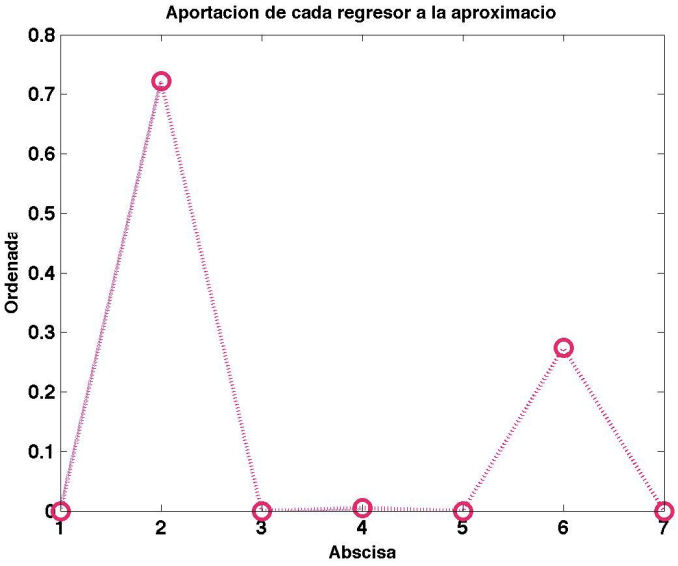


Figura 4.33: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 7, y la ordenada es la contribución a la salida (prueba 1).

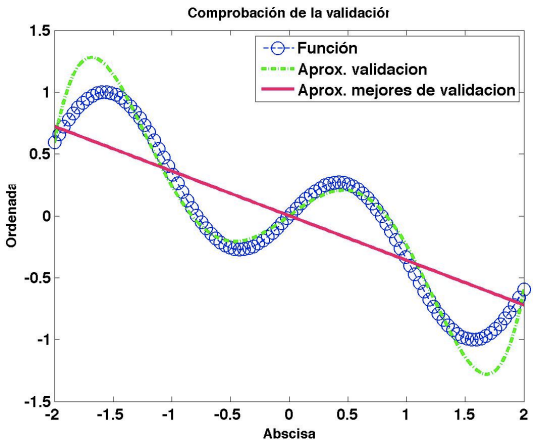


Figura 4.34: Con una neurona (prueba 1).

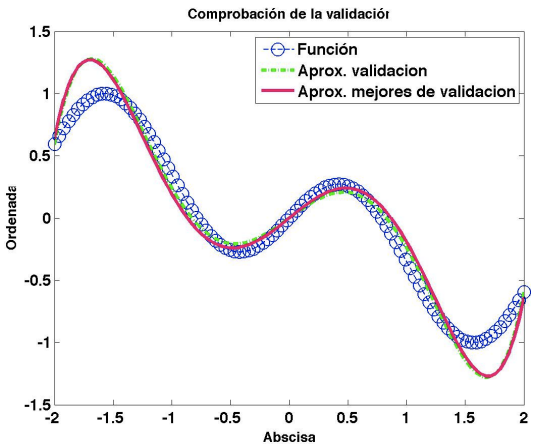


Figura 4.35: Con dos neuronas (prueba 2).

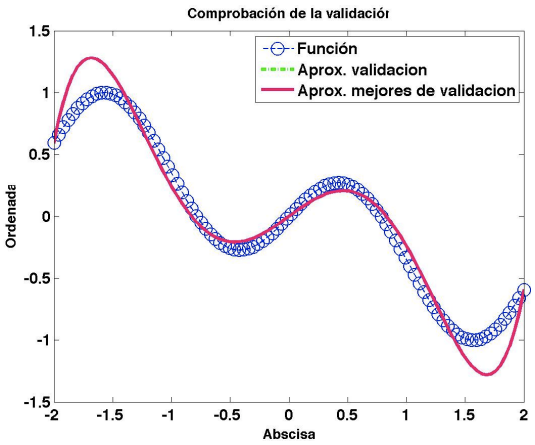


Figura 4.36: Con tres neuronas (prueba 3).

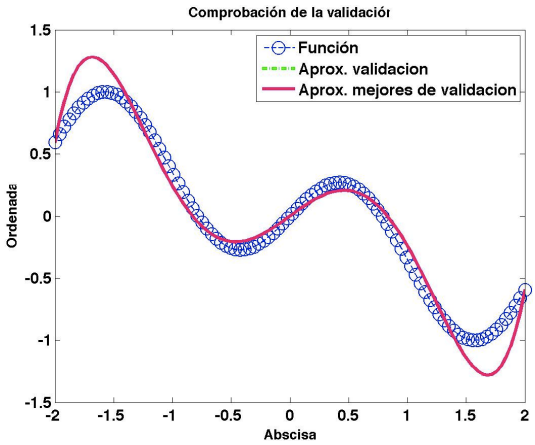


Figura 4.37: Con cuatro neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.10 en puntos regulares.

A continuación de muestran los datos de la aproximación de la función (3.10) en puntos regulares. Ésta función es mas difícil de aproximar, y por ello hemos elegido doce puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	-2	2	12	100	**1**
2	-2	2	12	100	**3**
3	-2	2	12	100	**5**
4	-2	2	12	100	**7**

Tabla 4.23: Datos de la prueba para estudiar el número de regresores necesarios para la función 3.10 en puntos regulares.

En la tabla 4.24, se muestra el error FVU y el porcentaje de energía para los datos de entrenamiento y validación.

PRUEBA	Error FVU entrenamiento (mejores regresores)	Error FVU validación (mejores regresores)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	7.661e-01	9.005e-01	1	0.282
2	2.628e-01	4.279e-01	1	0.754
3	2.069e-02	4.186e-02	1	0.981
4	4.782e-05	2.605e-04	1	0.999

Tabla 4.24 : Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.10 en puntos regulares.

Gráficas:

El mejor polinomio es de orden tres, que es el cuarto polinomio. Luego tenemos 5 polinomios que aportan bastante a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores(polinomios), la aproximación va a ir mejorando. En la función anterior, el error caía bruscamente de tener un polinomio a dos y luego prácticamente no descendía. En cambio, ahora el error de aproximación va a ir descendiendo constantemente. Efectivamente, este hecho se constata con los datos de la tabla 4.24 y las gráficas de la aproximación de validación de la prueba 1 a la 4.

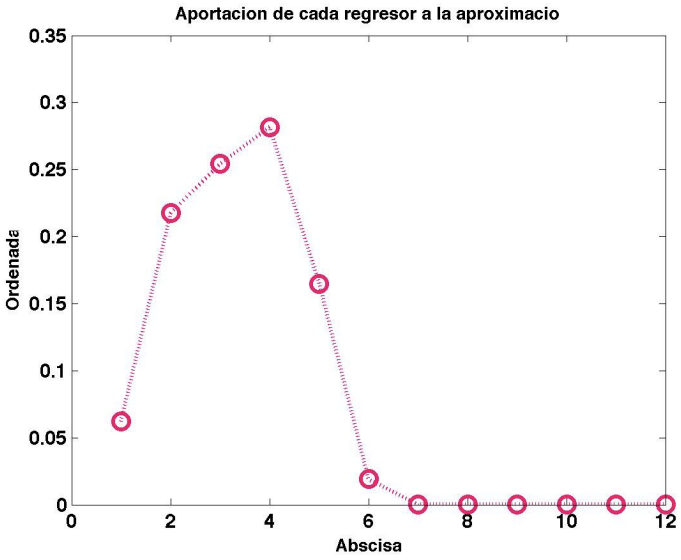


Figura 4.38: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 12, y la ordenada es la contribución a la salida (prueba 1).

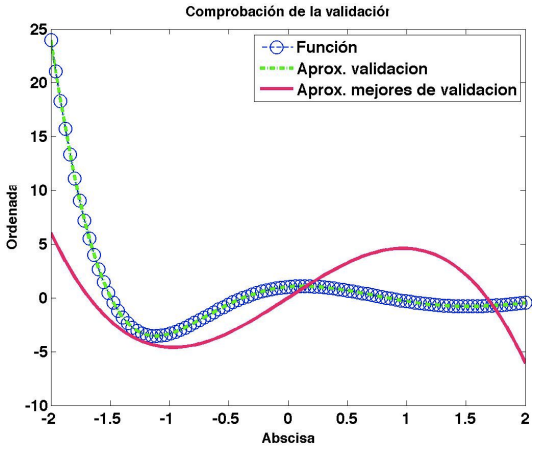


Figura 4.39: Con una neurona (prueba 1).

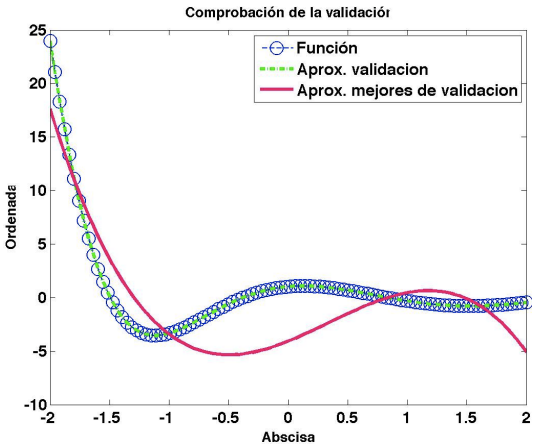


Figura 4.40: Con tres neuronas (prueba 2).

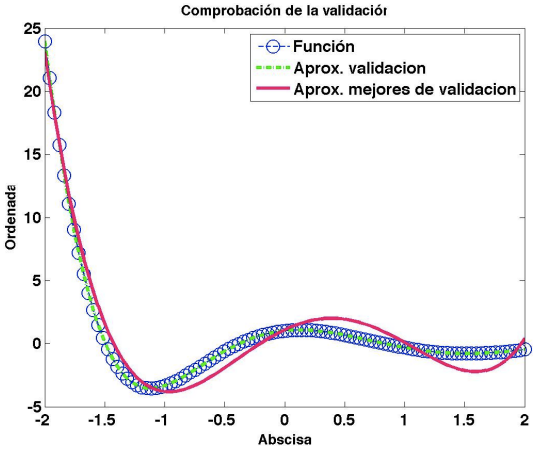


Figura 4.41: Con cinco neuronas (prueba 3).

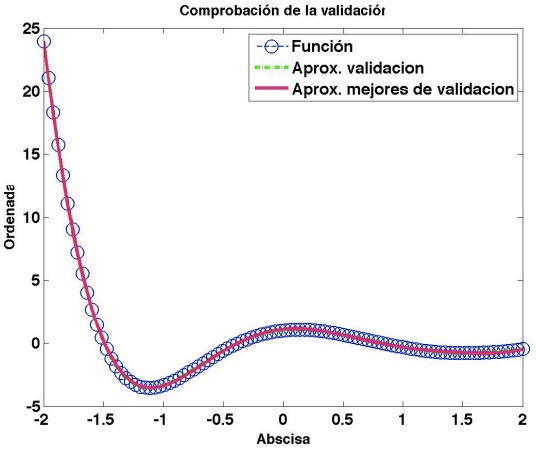


Figura 4.42: Con siete neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.11 en puntos regulares.

A continuación se muestran los datos de la aproximación de la función 3.11 en puntos regulares. Ésta función también es difícil de aproximar, y por ello hemos elegido quince puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	0	0.3	15	100	**1**
2	0	0.3	15	100	**4**
3	0	0.3	15	100	**7**
4	0	0.3	15	100	**10**

Tabla 4.25: Datos de la prueba para estudiar el número de regresores necesarios para la función 3.11 en puntos regulares.

PRUEBA	Error FVU entrenamiento (mejores regresores)	Error FVU validación (mejores regresores)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	6.861e-01	6.214e-01	1	0.341
2	1.376e-01	1.845e-01	1	0.869
3	7.409e-02	3.205e-01	1	0.964
4	1.232e-02	3.715e-01	1	1.024

Tabla 4.26: Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.11 en puntos regulares.

Gráficas:

De nuevo el mejor polinomio es de orden cuatro, que es el quinto polinomio. Luego tenemos siete polinomios que aportan bastante a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores(polinomios), la aproximación va a ir mejorando. En la función 3.9, el error caía bruscamente de tener un polinomio a dos y luego prácticamente no descendía. En cambio, ahora el error de aproximación va a ir descendiendo constantemente como ocurría en la función 3.10.

Efectivamente, este hecho se constata con los datos de la tabla 4.26 y las gráficas de la aproximación de validación de la prueba 1 a la 4.

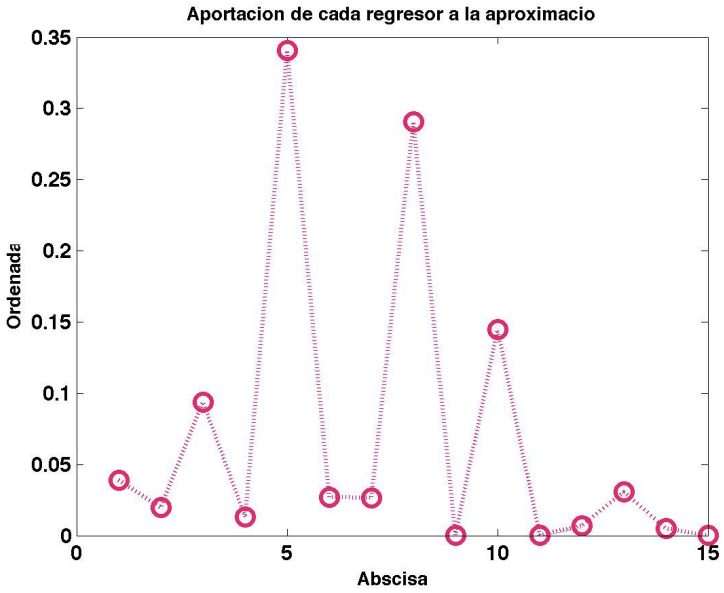


Figura 4.43: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 15, y la ordenada es la contribución a la salida (prueba 1).

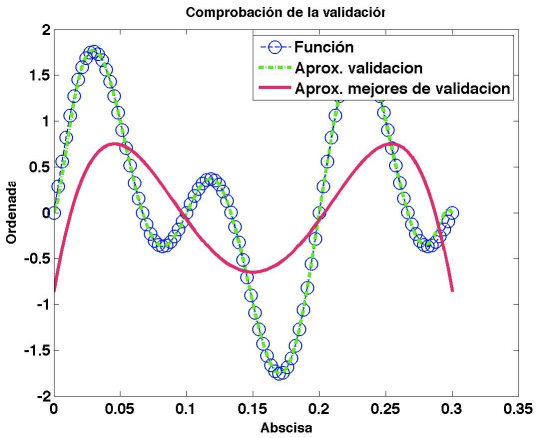


Figura 4.44: Con una neurona (prueba 1).

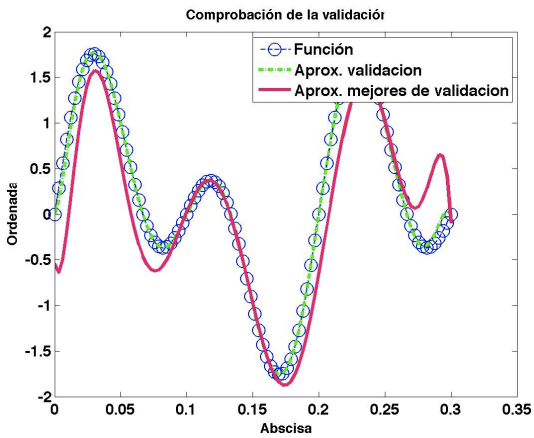


Figura 4.45: Con cuatro neuronas (prueba 2)**.**

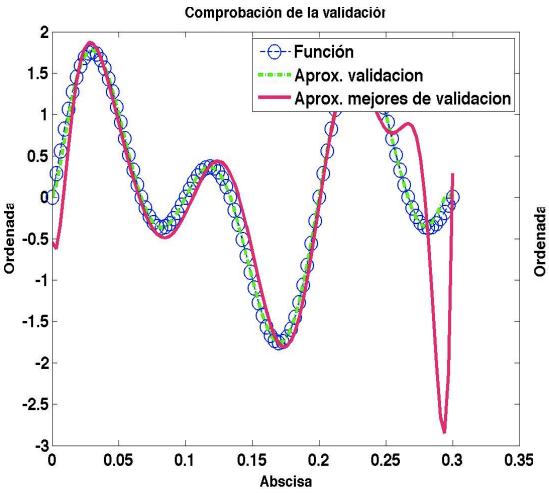


Figura 4.46: Con siete neuronas (prueba 3).

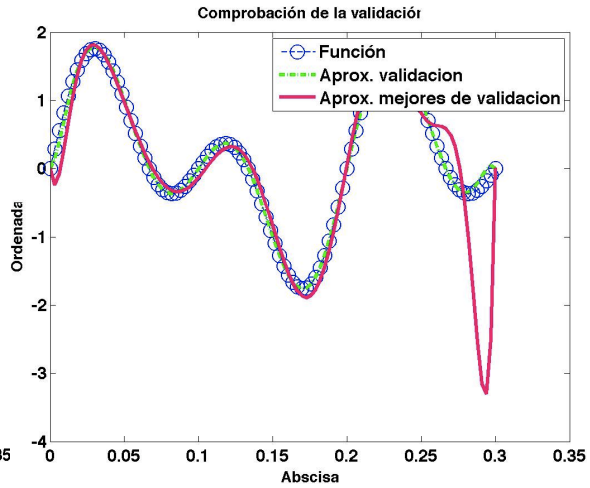


Figura 4.47: Con diez neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.9 en puntos arbitrarios.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	-2	2	7	100	1
2	-2	2	7	100	2
3	-2	2	7	100	3
4	-2	2	7	100	4

Tabla 4.27: Datos de la prueba para estudiar el número de regresores necesarios para la función 3.9 en puntos arbitrarios.

PRUEBA	Error FVU entrenamiento (mejores regresores)	Error FVU validación (mejores regresores)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	1.098e-01	2.856e-01	1	0.867
2	8.731e-01	2.969e+00	1	2.152
3	9.961e-01	1.247e+00	1	1.254
4	2.796e+01	2.135e+01	1	18.669

Tabla 4.28: Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.9 en puntos arbitrarios.

Gráficas:

El mejor polinomio es el de orden uno, que es el segundo polinomio. Luego tenemos un polinomio que aporta poco a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores (polinomios), la aproximación va a ir mejorando.

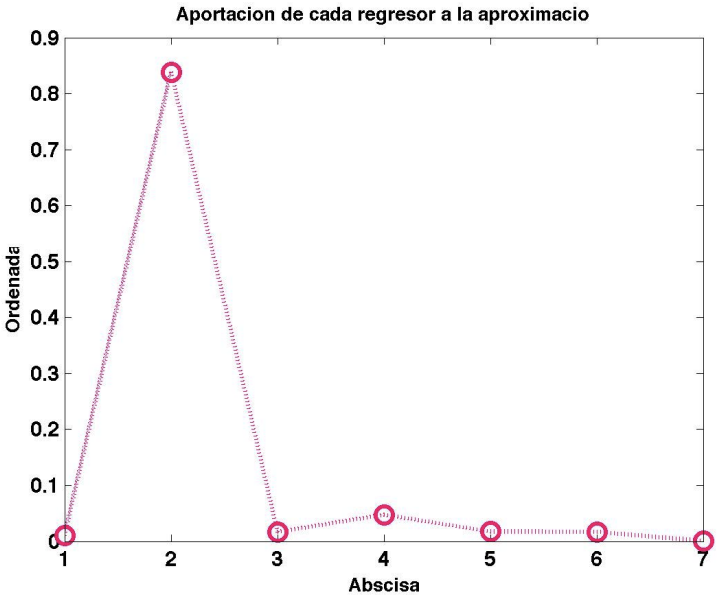


Figura 4.48: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 7, y la ordenada es la contribución a la salida (prueba 1).

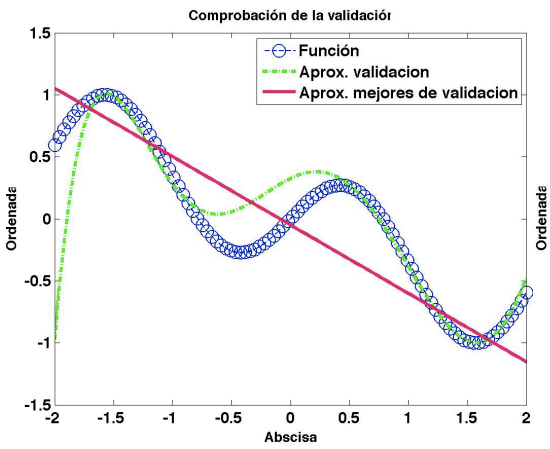


Figura 4.49: Con una neurona (prueba 1).

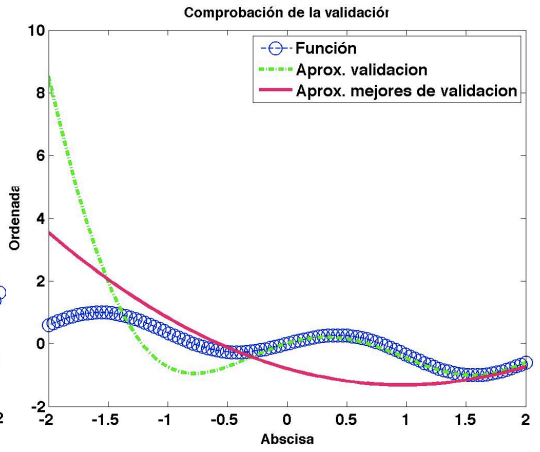


Figura 4.50: Con dos neuronas (prueba 2).

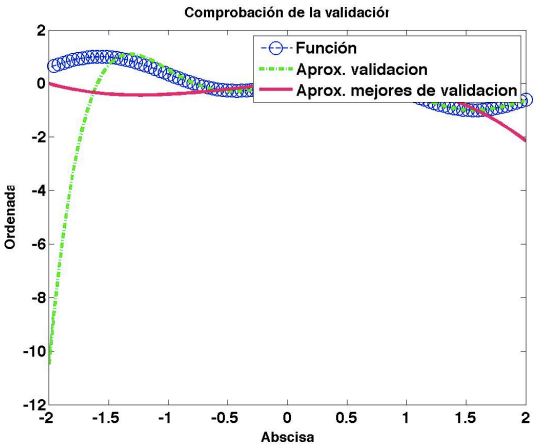


Figura 4.51: Con tres neuronas (prueba 3).

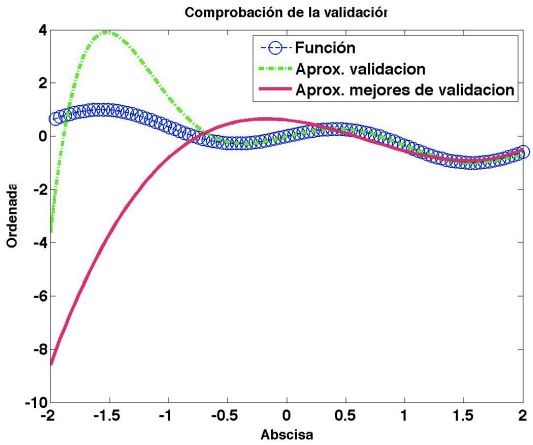


Figura 4.52: Con cuatro neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.10 en puntos arbitrarios.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	-2	2	12	100	**1**
2	-2	2	12	100	**3**
3	-2	2	12	100	**5**
4	-2	2	12	100	**7**

Tabla 4.29: Datos de la prueba para estudiar el número de regresores necesarios para la función 3.10 en puntos arbitrarios.

PRUEBA	Error FVU entrenamiento (mejores regresores)	Error FVU validación (mejores regresores)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	7.968e-01	1.014e+00	1	0.299
2	2.845e-01	6.406e-01	1	0.749
3	1.538e-02	4.046e-02	1	0.986
3	8.547e-05	1.899e-04	1	0.999

Tabla 4.30: Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.10 en puntos arbitrarios.

Gráficas:

El mejor polinomio es el de orden uno, que es el segundo polinomio. Luego tenemos cuatro polinomios que aportan bastante a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores (polinomios), la aproximación va a ir mejorando.

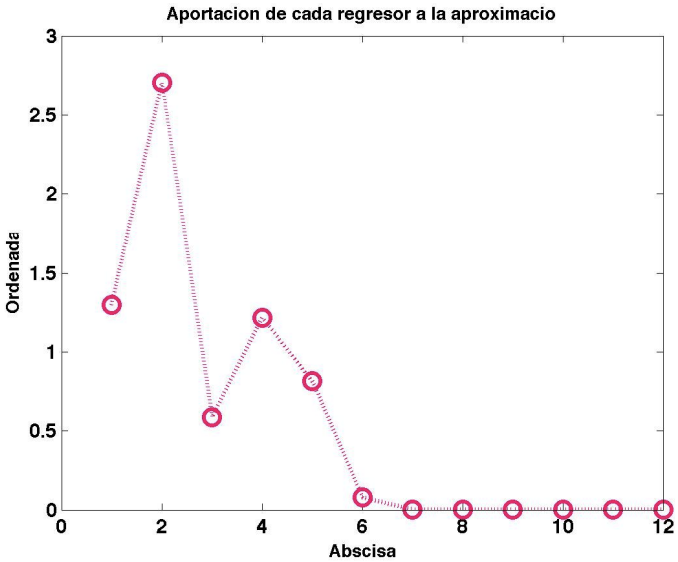


Figura 4.53: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 12, y la ordenada es la contribución a la salida (prueba 1).

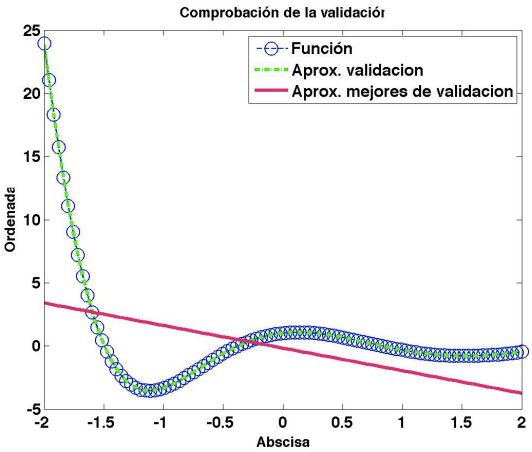


Figura 4.54: Con una neurona (prueba 1).

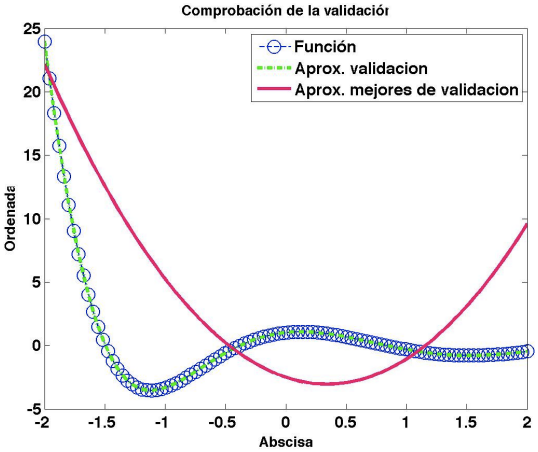


Figura 4.55: Con tres neuronas (prueba 2).

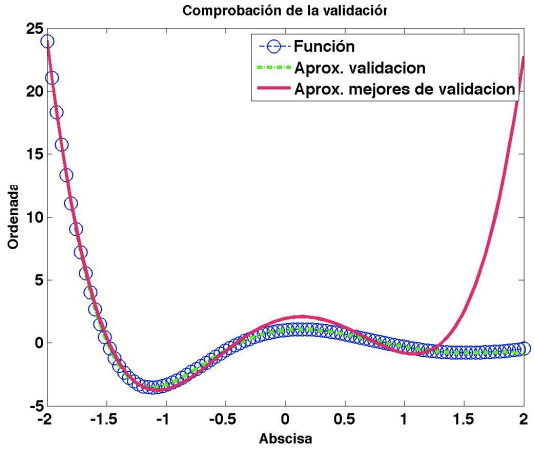


Figura 4.56: Con cinco neuronas (prueba 3).

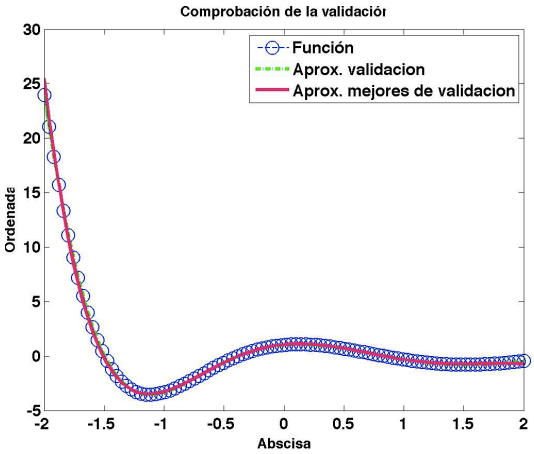


Figura 4.57: Con siete neuronas (prueba 4).

- Tabla y gráficas del estudio de la función 3.11 en puntos arbitrarios.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas
1	0	0.3	15	100	**1**
2	0	0.3	15	100	**4**
3	0	0.3	15	100	**7**
4	0	0.3	15	100	**10**

Tabla 4.31: Datos de la prueba para estudiar el número de regresores necesarios para la función 3.11 en puntos arbitrarios.

PRUEBA	Error FVU entrenamiento (mejores regresores)	Error FVU validación (mejores regresores)	Evolución porcentaje energía (entrenamiento)	Evolución porcentaje energía (validación)
1	5.733e-01	1.683e+00	1	0.449
2	5.886e-02	1.951e+02	1	0.933
3	1.636e-01	2.878e+03	1	1.086
4	7.867e-01	5.147e+03	1	1.952

Tabla 4.32: Parámetros de error de entrenamiento y validación de las mejores neuronas, y porcentaje de energía de entrenamiento y validación de la función 3.11 en puntos arbitrarios.

Gráficas:

El mejor polinomio es el de orden tres, que es el cuarto polinomio. Luego tenemos cinco polinomios que aportan bastante a la función de la aproximación deseada. Esto significa, que conforme vayamos añadiendo regresores (polinomios), la aproximación va a ir mejorando.

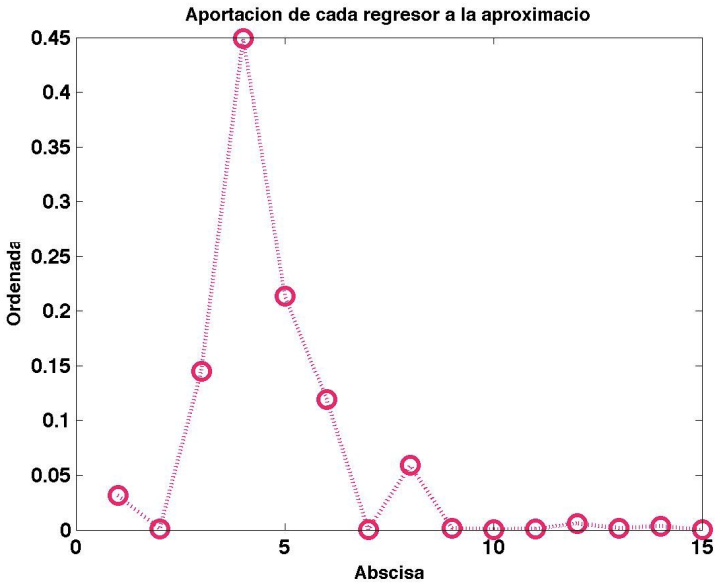


Figura 4.58: Aquí la abscisa es el número del polinomio, que en este caso va de 1 a 15, y la ordenada es la contribución a la salida (prueba 1).

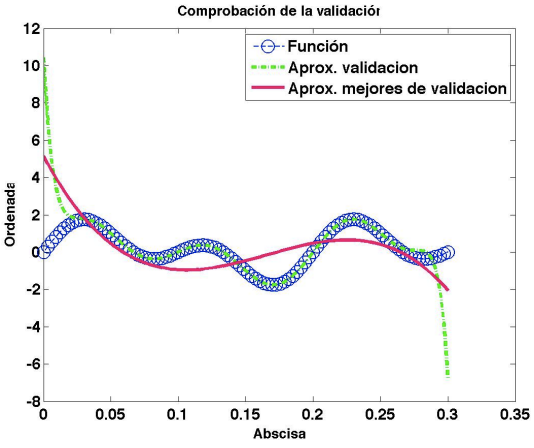


Figura 4.59: Con una neurona (prueba 1).

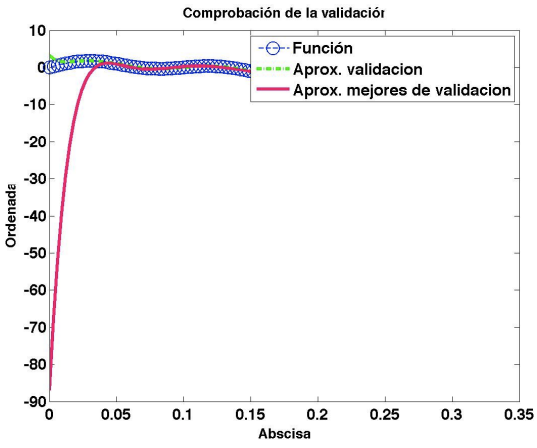


Figura 4.60: Con cuatro neuronas (prueba 2).

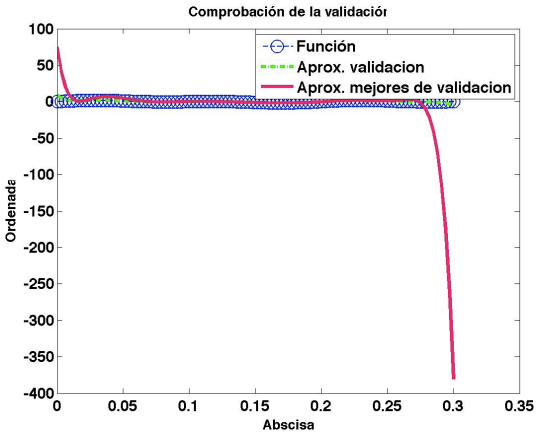


Figura 4.61: Con siete neuronas (prueba 3).

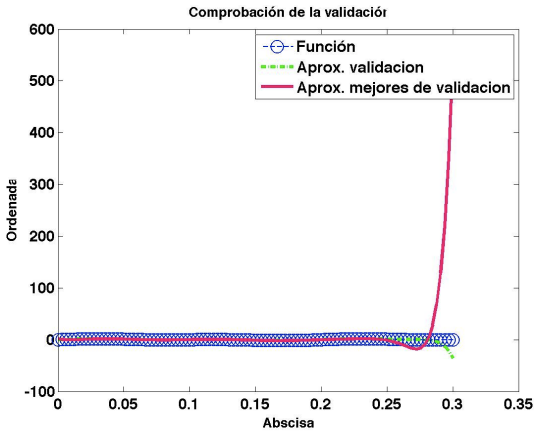


Figura 4.62: Con diez neuronas (prueba 4).

Para finalizar este apartado, podríamos decir que tanto para puntos regulares como arbitrarios, se observa que el error FVU disminuye cada vez más y que el porcentaje de la energía de la función deseada, expresada por la red neuronal compuesta por los mejores regresores, cada vez aumenta más .

También, la red neuronal selecciona los regresores que más contribuyen a la salida deseada (la función que queremos aproximar).

4.2.4. Estudio del comportamiento de la red neuronal en presencia de ruido en los datos de entrenamiento. En puntos regulares y arbitrarios.

Para finalizar este capítulo, estudiamos por separado las tres funciones y observamos qué ocurre en la aproximación cuando los datos de entrenamiento están contaminados con ruido. Para ello mostraremos el error de aproximación y las gráficas de entrenamiento y validación para diferentes número de puntos de entrenamiento, de regresores y diferentes niveles de ruido (SNR). Hay que destacar que el ruido es de tipo blanco gaussiano.

Como a las tres funciones le sucede lo mismo, finalizaremos con una única conclusión.

- Tabla y gráficas del estudio de la función 3.9 en puntos regulares.

La tabla 4.33 se muestran los datos de las pruebas realizadas en la función 3.9 cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	-2	2	12	100	8	**10**
2	-2	2	12	100	8	**15**

Tabla 4.33: Estudio de la función 3.9 en puntos regulares cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

A continuación, mostramos los errores de aproximación y validación para las pruebas anteriores. Las dos primeras columnas corresponden a la red neuronal compuesta con todos los posibles polinomios, y las dos últimas columnas, a la red neuronal que está compuesta por las mejores neuronas.

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	2.927e-32	2.626e-01	13.848e-03	5.274e-01
2	8.494e-32	7.069e+00	2.285e-03	7.121e+00

Tabla 4.34: Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.9 en puntos regulares.

Gráficas:

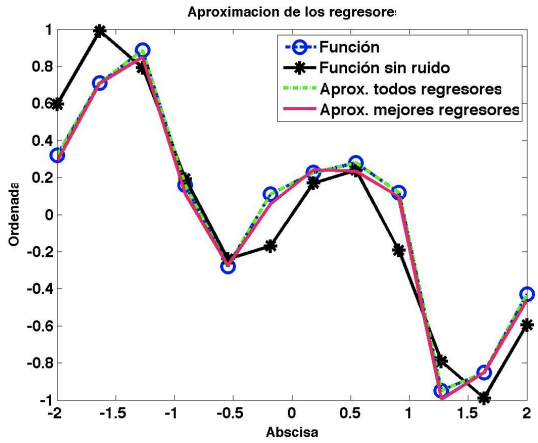


Figura 4.63: Aproximación de entrenamiento con la función con ruido (prueba 1).

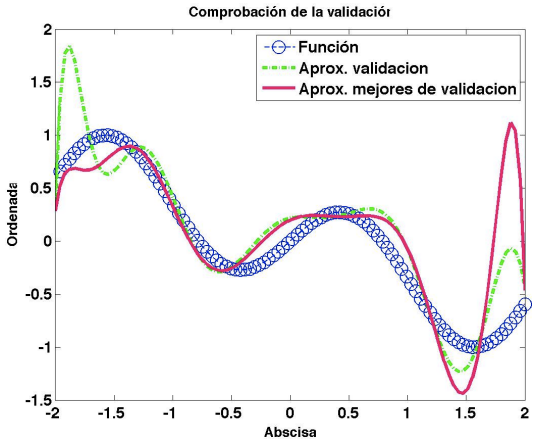


Figura 4.64: Aproximación de validación (prueba 1).

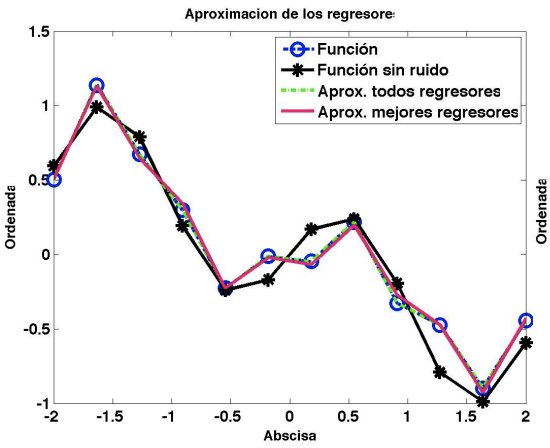


Figura 4.65: Aproximación de entrenamiento con la función con ruido (prueba 2).

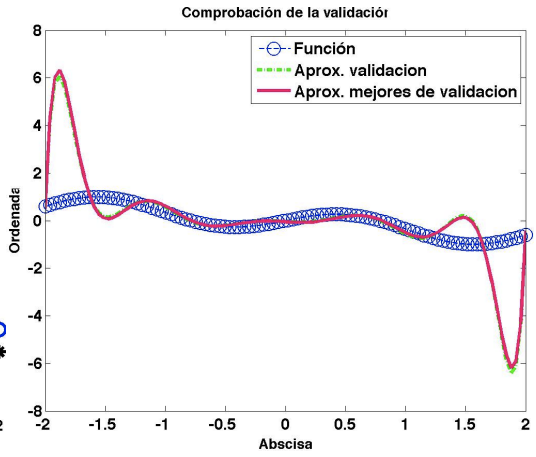


Figura 4.66: Aproximación de validación (prueba 2).

- Tabla y gráficas del estudio de la función 3.10 en puntos regulares.

En la tabla 4.35 se muestran los datos de las pruebas realizadas en la función 3.10 cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	-3	3	20	100	16	**10**
2	-3	3	20	100	16	**15**

Tabla 4.35: Estudio de la función 3.10 en puntos regulares cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	2.281e-31	5.379e+03	9.448e-04	3.376e+03
2	3.038e-31	2.129e+04	3.697e-03	2.140e+04

Tabla 4.36: Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.10 en puntos regulares.

Gráficas:

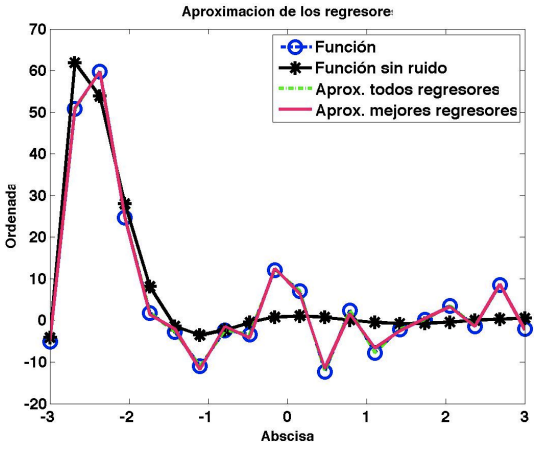


Figura 4.67: Aproximación de entrenamiento con la función con ruido (prueba 1)**.**

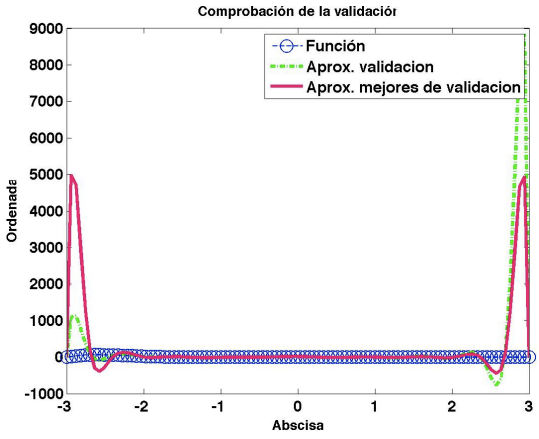


Figura 4.68: Aproximación de validación (prueba 1)**.**

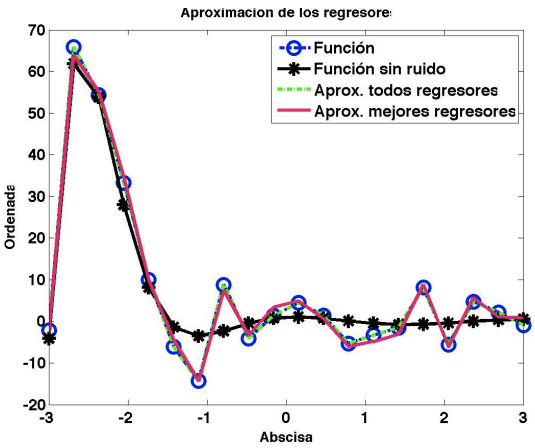


Figura 4.69: Aproximación de entrenamiento con la función con ruido (prueba 2)**.**

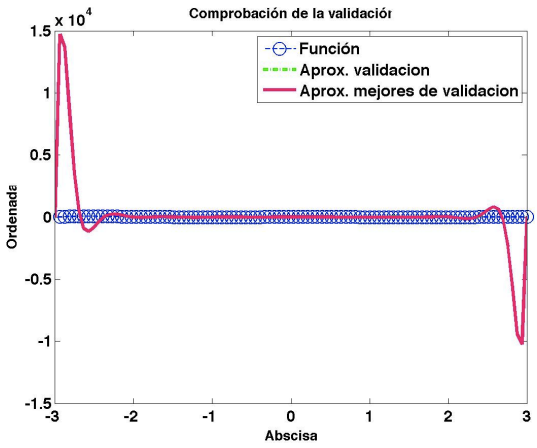


Figura 4.70: Aproximación de validación (prueba 2)**.**

- Tabla y gráficas del estudio de la función 3.11 en puntos regulares.

En la tabla 4.37 se muestran los datos de las pruebas realizadas en la función 3.11 cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	0	0.3	20	100	16	**10**
2	0	0.3	20	100	16	**15**

Tabla 4.37: Estudio de la función 3.11 en puntos regulares cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	5.199e-19	9.567e+03	7.185e-01	9.608e+03
2	7.641e-18	6.115e+04	8.908e-01	6.128e+04

Tabla 4.38: Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.11 en puntos regulares.

Gráficas:

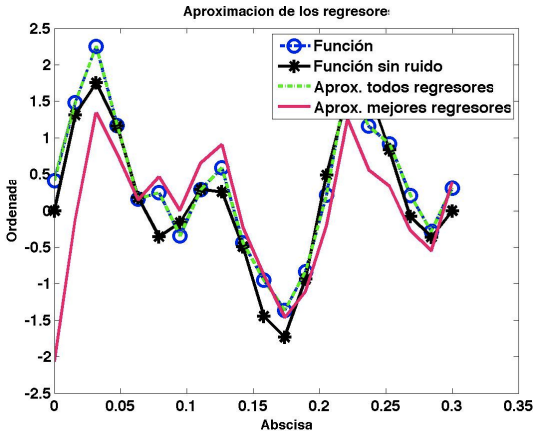


Figura 4.71: Aproximación de entrenamiento con la función con ruido (prueba 1)**.**

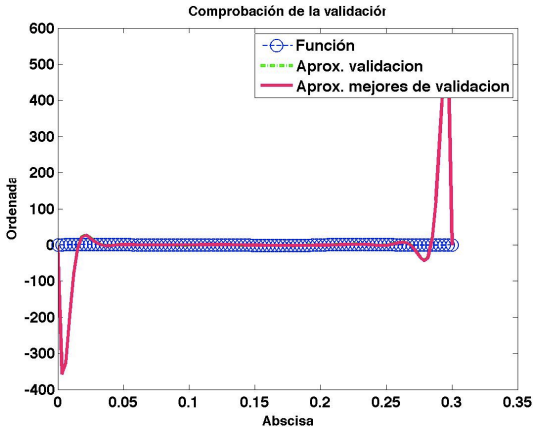


Figura 4.72: Aproximación de validación (prueba 1)**.**

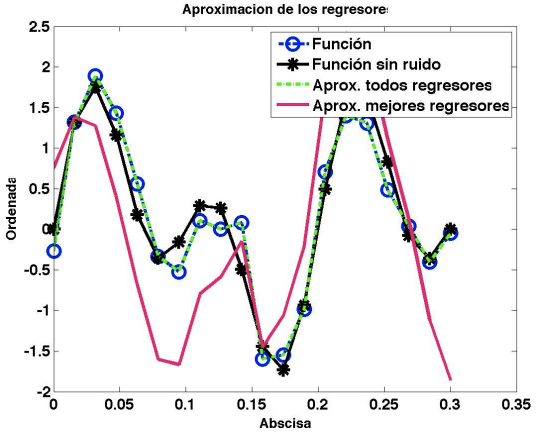


Figura 4.73: Aproximación de entrenamiento con la función con ruido (prueba 2)**.**

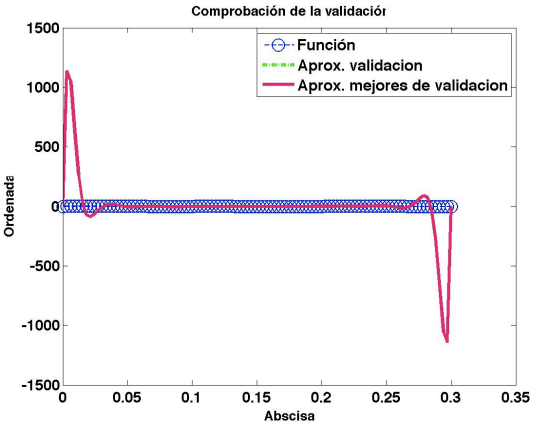


Figura 4.74: Aproximación de validación (prueba 2)**.**

- Tabla y gráficas del estudio de la función 3.9 en puntos aleatorios.

La tabla 4.39 se muestran los datos de las pruebas realizadas en la función 3.9 con puntos aleatorios cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	-2	2	12	100	8	**10**
2	-2	2	12	100	8	**15**

Tabla 4.39: Estudio de la función 3.9 en puntos aleatorios cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	3.139e-13	3.967e+17	3.622e+16	4.323e+17
2	1.730e-31	5.279e-01	7.419e-04	4.980e-01

Tabla 4.40 : Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.9 en puntos aleatorios.

Gráficas:

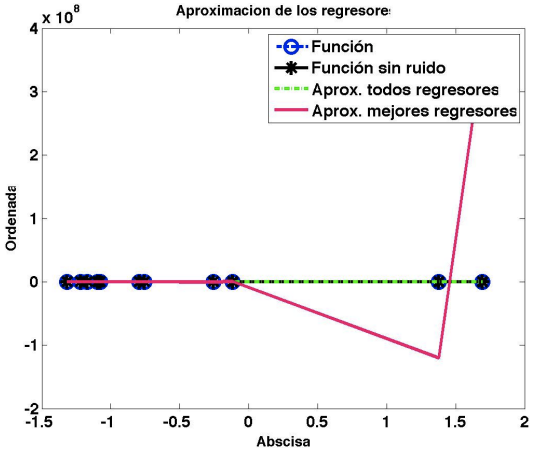


Figura 4.75: Aproximación de entrenamiento con la función con ruido (prueba 1)**.**

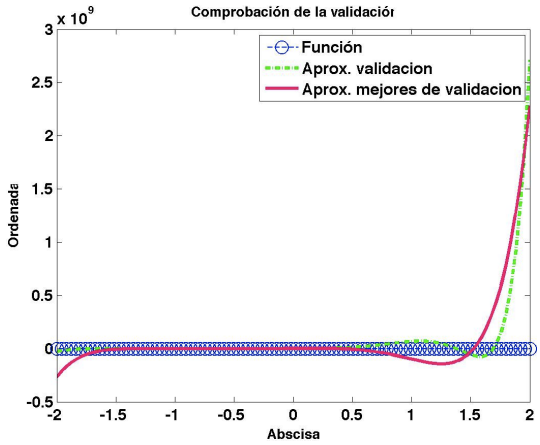


Figura 4.76: Aproximación de validación (prueba 1)**.**

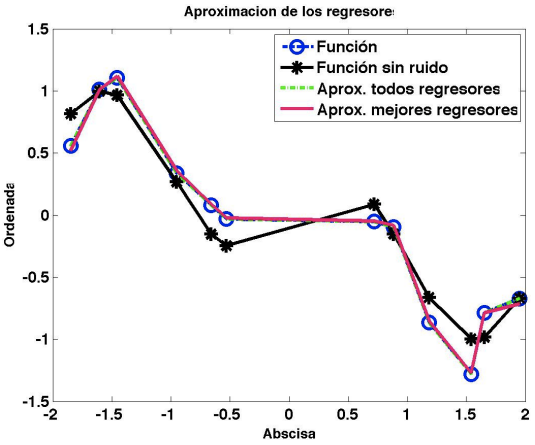


Figura 4.77: Aproximación de entrenamiento con la función con ruido (prueba 2)**.**

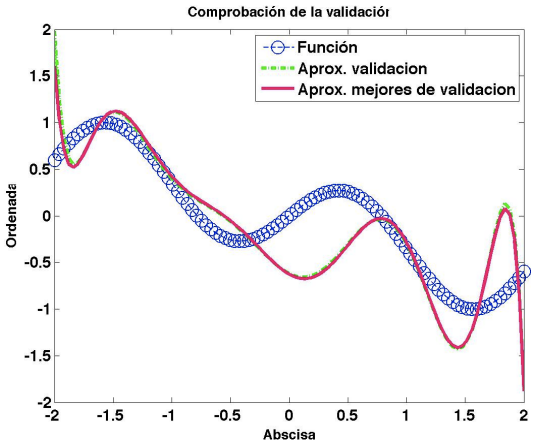


Figura 4.78: Aproximación de validación (prueba 2)**.**

- Tabla y gráficas del estudio de la función 3.10 en puntos aleatorios.

La tabla 4.41 se muestran los datos de las pruebas realizadas en la función 3.10 con puntos aleatorios cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	-3	3	20	100	16	**10**
2	-3	3	20	100	16	**15**

Tabla 4.41: Estudio de la función 3.10 en puntos aleatorios cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	5.013e-24	3.495e+11	1.727e+07	1.938e+13
2	2.230e-17	3.725e+16	5.993e+08	3.052e+16

Tabla 4.42 : Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.10 en puntos aleatorios.

Gráficas:

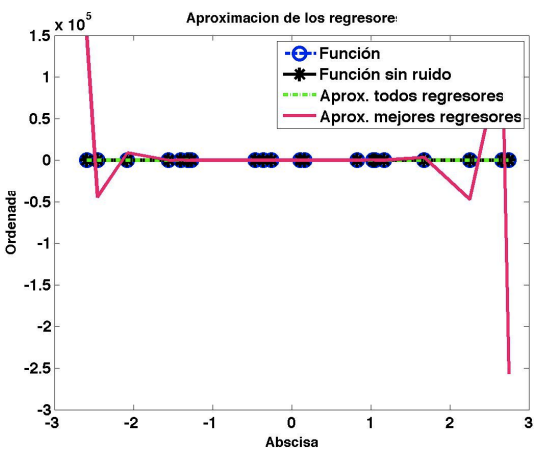


Figura 4.79: Aproximación de entrenamiento con la función con ruido (prueba 1).

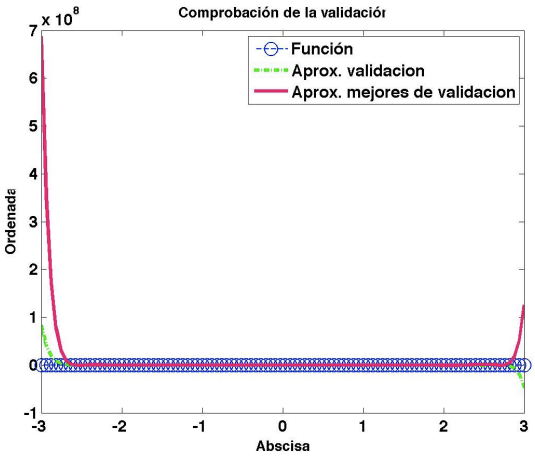


Figura 4.80: Aproximación de validación (prueba 1).

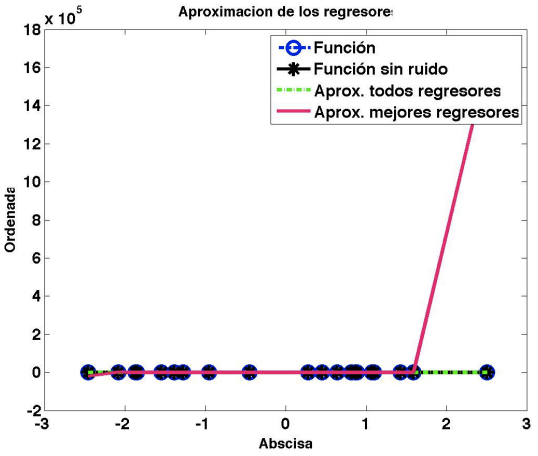


Figura 4.81: Aproximación de entrenamiento con la función con ruido (prueba 2).

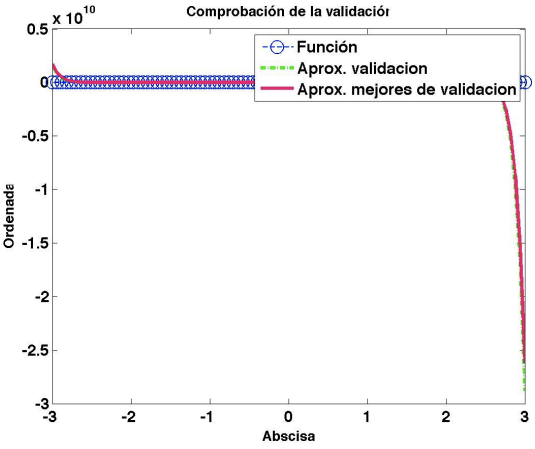


Figura 4.82: Aproximación de validación (prueba 2).

- Tabla y gráficas del estudio de la función 3.11 en puntos aleatorios.

La tabla 4.43 se muestran los datos de las pruebas realizadas en la función 3.11 con puntos aleatorios cuando añadimos ruido a los puntos de entrenamiento.

PRUEBA	Inicial	Final	n	n val	Número mejores neuronas	SNR
1	0	0.3	20	100	16	**10**
2	0	0.3	20	100	16	**15**

Tabla 4.43: Estudio de la función 3.11 en puntos aleatorios cuando añadimos ruido a los puntos de entrenamiento (ruido de tipo blanco gaussiano).

PRUEBA	Error FVU entrenamiento (todos los polinomios)	Error FVU validación (todos los polinomios)	Error FVU entrenamiento (mejores polinomios)	Error FVU validación (mejores polinomios)
1	1.508e-13	8.709e+09	3.773e+09	8.663e+09
2	7.832e-15	2.650e+11	2.044e+08	2.511e+11

Tabla 4.4 : Parámetros de error de entrenamiento y validación de todas las neuronas y de las mejores neuronas de la función 3.11 en puntos aleatorios.

Gráficas:

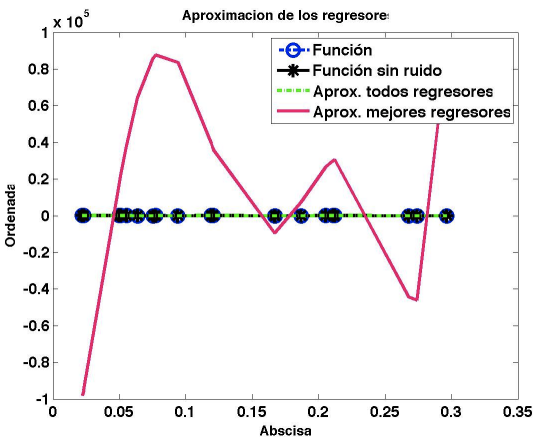


Figura 4.83: Aproximación de entrenamiento con la función con ruido (prueba 1)**.**

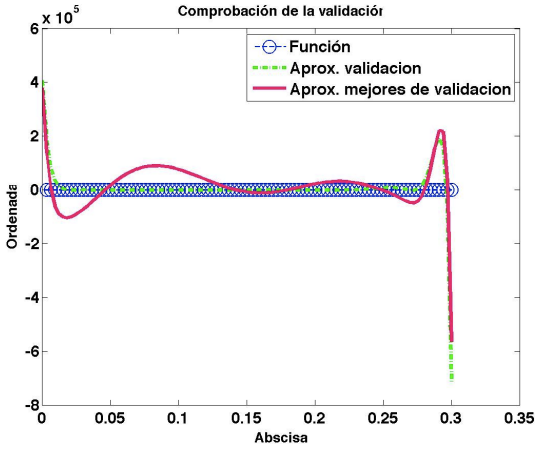


Figura 4.84: Aproximación de validación (prueba 1)**.**

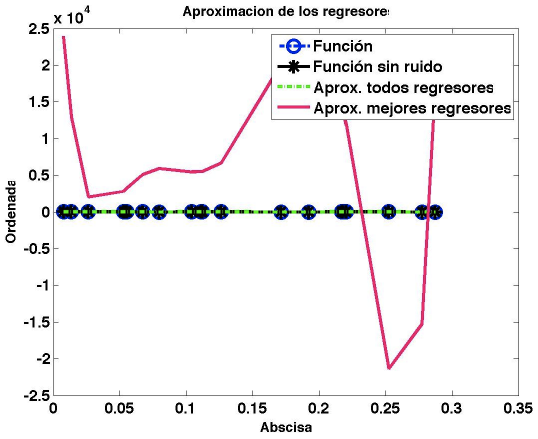


Figura 4.85: Aproximación de entrenamiento con la función con ruido (prueba 2)**.**

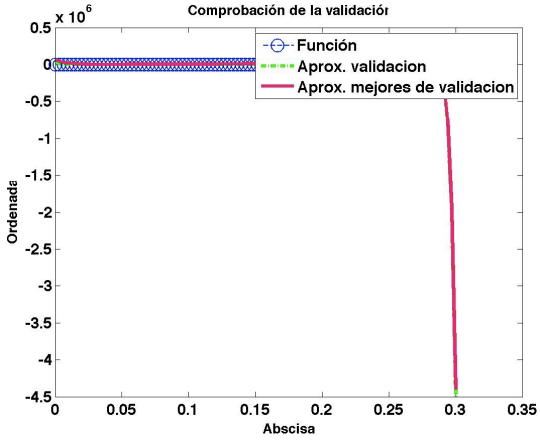


Figura 4.86: Aproximación de validación (prueba 2)**.**

Lo que queremos mostrar en estas gráficas y tablas, tanto en puntos regulares como aleatorios, es que el error de aproximación de validación es, lógicamente, mucho peor cuando existe ruido en los puntos de entrenamiento. Como se ve en las gráficas de aproximación de los puntos de entrenamiento la red neuronal "aprende" de unos datos que no son los de la función (ya que tienen ruido). Así, la red no generaliza bien, la salida de los datos de validación difiere mucho de la función deseada. Cuando mayor sea el ruido (menor SNR), peor aproxima la red neuronal.

5. Redes neuronales de Chebychev multidimensionales.

La Red neuronal de Chebychev realiza una transformación no lineal entre un espacio de dimensión n y un espacio de salida unidimensional: $\Re^n \rightarrow \Re$. Así, los datos de entrada son vectores de dimensión $n : x = \{x_1, x_2, \ldots, x_n\}$. Cada coordenada pertenece al rango [−1, 1] de forma que se cumple: $x \in [-1,1]^n$. Al igual que en el espacio unidimensional, si la variable independiente de la función original pertenece a un rango distinto de [−1, 1] se lleva a cabo un cambio de variable para trasladar cada coordenada al intervalo de trabajo:

$$x_m = \frac{x_{m_{or}} - \frac{1}{2}\left(x_{m_{or\ \sup}} + x_{m_{or\ \inf}}\right)}{\frac{1}{2}\left(x_{m_{or\ \sup}} - x_{m_{or\ \inf}}\right)} \quad Para\ 1 \le m \le n \tag{5.1}$$

donde $x_{m_{or\ \inf}}$ y $x_{m_{or\ \sup}}$ son los límites inferior y superior originales de la coordenada *m*-ésima, $x_{m_{or}}$ es el valor de la coordenada en el rango original y x_m es el valor de la coordenada en el rango de los polinomios de Chebychev [−1, 1].

Para conseguir un conjunto de regresores ortogonales en el espacio multidimensional $\Re^n$ debemos definir una función neuronal basada en polinomios de Chebychev. Además, se debe aplicar un muestreo adecuado en cada una de las coordenadas. Escogemos como función neuronal el producto de *n* polinomios de Chebychev:

$$P_i(x) = T_{0_1(i)}(x_1) T_{0_2(i)}(x_2) \cdots T_{0_n(i)}(x_n) \tag{5.2}$$

Como se observa en la ecuación 5.2, el polinomio *i*-ésimo toma como variable independiente el valor de la coordenada *i*-ésima. Cada polinomio tiene un orden particular, cada neurona tiene una función neuronal única, es decir, un producto de polinomios diferente del resto de productos. De esta forma se posibilita una gran variedad de formas en la función neuronal.

Para completar la descripción de las Redes neuronales de Chebychev se requiere la definición de un muestreo de entrenamiento que posibilite la ortogonalidad de los regresores durante dicha fase. Para ello, la coordenada *m*-ésima será muestreada en las raíces del polinomio de orden superior *p*(*m*). Si, por ejemplo, se desea que la coordenada *m*-ésima sea muestreada en diez puntos, entonces el polinomio de orden superior será *p*(*m*) = 10, los puntos de entrenamiento serán las raíces del polinomio de Chebychev de orden diez. La función neuronal tendrá diez posibles opciones en la coordenada *m* correspondientes a los polinomios con orden comprendido entre 0 y *p*(*m*) − 1 = 9. Este procedimiento ampliado a todas las coordenadas genera una rejilla regular o cubo multidimensional de puntos. En esta rejilla cada lado representa una

dimensión o coordenada. Además, cada coordenada del espacio de entrada es muestreada con un número particular de raíces. Así, el polinomio de orden superior de la coordenada *r*, *p*(*r*), puede ser distinto de *p*(*m*) si $r \neq m$. El esquema en forma de rejilla produce un número de datos de entrenamiento igual a *N* = *p*(1)*p*(2)...*p*(*m*)...*p*(*n*). Los puntos de muestreo en la coordenada *m*-ésima son las raíces expresadas como:

$$x_{k_m(z)} = \cos\left(\frac{\pi(k_m(z)-1/2)}{p(m)}\right) \tag{5.3}$$

donde *m* = 1, 2, . . . , *n*. El índice $k_m(z)$ pertenece al conjunto de valores {1, 2, . . . , *p*(*m*)}, donde z es un índice que varía desde 1 hasta el número total de datos *N*. Cada dato de entrenamiento se representa como un vector:

$$x_{k(z)} = \left\{x_{k_1(z)}, x_{k_2(z)}, \ldots, x_{k_n(z)}\right\} \tag{5.4}$$

El número máximo de neuronas es igual al número de datos de entrenamiento. Así mismo, el muestreo efectuado para entrenar la red neuronal impone una limitación en el orden de los polinomios de Chebychev que forman las funciones neuronales. Si la coordenada *m*-ésima es muestreada con *p*(*m*) puntos, el polinomio correspondiente a dicha coordenada en la neurona i-ésima tendrá como orden máximo $p(m)-1(o_m(i) < p(m))$. Si evaluamos la función neuronal de la neurona *i*-ésima en un dato de entrada cualquiera (no de entrenamiento) de la red neuronal **x** obtenemos:

$$P_i(x) = T_{0_1(i)}(x_1)T_{0_2(i)}(x_2)\cdots T_{0_n(i)}(x_n) \;\; donde \;\; 0_1(i) \in \left\{0,1,2,\ldots p(1)-1\right\}$$

$$0_2(i) \in \left\{0,1,2,\ldots p(2)-1\right\}$$

$$\vdots$$

$$0_n(i) \in \left\{0,1,2,\ldots p(n)-1\right\} \tag{5.5}$$

La limitación en el orden de los polinomios de cada coordenada está determinada por la propiedad de ortogonalidad denominada:

$$\sum_{k=1}^{N} T_i(x_k)T_j(x_k) = \begin{cases} 0 & i \neq j \\ \dfrac{m}{2} & i = j \neq 0 \\ m & i = j = 0 \end{cases} \tag{5.6}$$

La ortogonalidad de los regresores en el espacio de entrada multidimensional deberá cumplir la siguiente condición:

$$P_iP_j = P_i(x_k)^T P_j(x_k) = \sum_{z=1}^{N} P_i(x_{k(z)})P_j(x_{k(z)}) \begin{cases} 0 & i \neq j \\ \neq 0 & i = j \end{cases} \tag{5.7}$$

donde $P_i(x_k)$ y $P_j(x_k)$ son los regresores fruto de evaluar respectivamente las funciones neuronales P_i y P_j en el conjunto de entrenamiento $x_k = x_{k(1)}, x_{k(2)}, \ldots, x_{k(N)}$.

A continuación, vamos a mostrar las cuatro primeres funciones del conjunto de entrenamiento. Las funciones constan de 225 muestras distribuidas en una rejilla de puntos. Como conjunto de validación se utilizaron 10000 muestras generadas de manera uniforme en el dominio $[0,1]^2$. Hay que tener en cuenta que los puntos de cada coordenada en el conjunto de entrenamiento en la Red neuronal de Chebychev deben corresponder a las raíces de polinomios de Chebychev de orden superior de la respectiva coordenada. En cambio, en el conjunto de validación no existe la necesidad de muestrear en las raíces de los polinomios de Chebychev y las muestras se encuentran equi-espaciadas. En esta red neuronal se utilizó una rejilla uniforme de puntos equi-espaciados en el conjunto de validación. Los puntos de validación se ubican en posiciones aproximadamente intermedias (intercaladas) respecto a los puntos de entrenamiento.

- Función AF(función aditiva):

$$f(x_1,x_2) = 1{,}3356\left\{1{,}5(1-x_1) + e^{2x_1-1}\sin(3\pi(x_1-0{,}6)^2) + e^{3(x_2-0{,}5)}\sin(4\pi(x_2-0{,}9)^2)\right\} \quad (5.8)$$

- Función RF(función radial):

$$f(x_1,x_2) = 24{,}234\left\{\left[(x_1-0{,}5)^2 + (x_2-0{,}5)^2\right]\left[0{,}75 - (x_1-0{,}5)^2 - (x_2-0{,}5)^2\right]\right\} \quad (5.9)$$

- Función SIF(función de interacción simple):

$$f(x_1,x_2) = 10{,}391\left\{(x_1-0{,}4)(x_2-0{,}6) + 0{,}36\right\} \quad (5.10)$$

- Función CIF(función de interacción complicada):

$$f(x_1,x_2) = 1{,}9\left\{1{,}35 + e^{(x_1-x_2)}\sin(13(x_1-0{,}6)^2)\sin(7x_2)\right\} \quad (5.11)$$

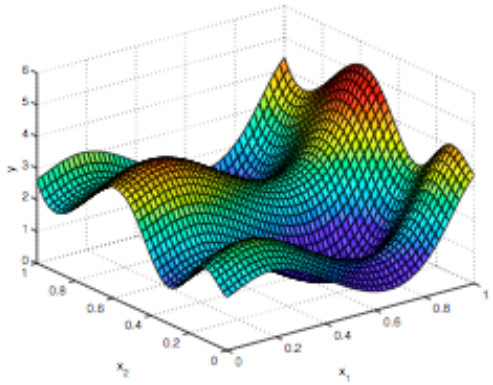


Figura AF

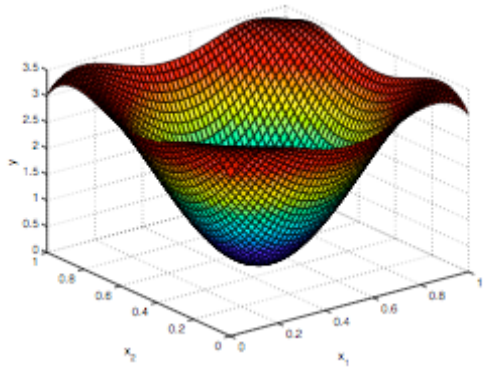


Figura RF

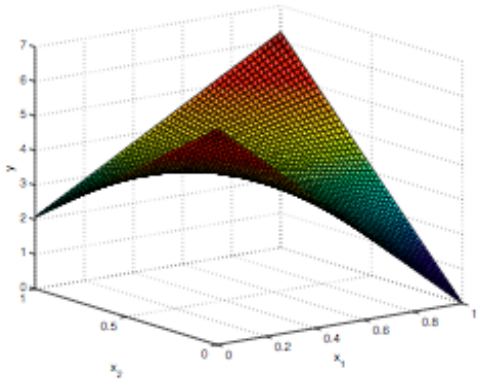


Figura SIF

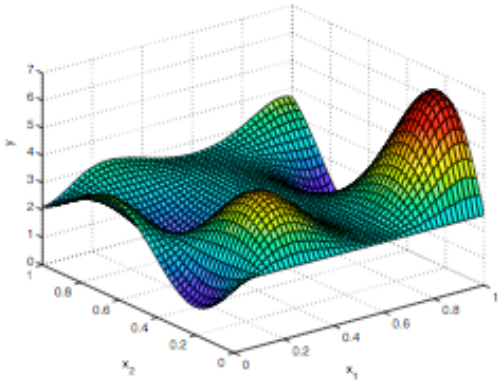


Figura CIF

Figuras 5.1: Funciones utilizadas en el test numérico de aproximación de funciones multidimensionales.

5.1. Estudio del comportamiento de la red neuronal sin presencia de ruido en los datos de entrenamiento.

En este apartado, vamos a estudiar el comportamiento de la red neuronal sin presencia de ruido. Para ello vamos a mostrar tablas de error FVU tanto en datos de entrenamiento como de validación, con pruebas de 5 hasta 30 neuronas, en intervalos de 5 en 5.

Las funciones poseen un intervalo en el espacio de entrada $[0,1]^2$; el numero de puntos de entrenamiento es de 10 puntos en cada dimensión, luego tenemos 100 puntos en total. El numero de puntos de validación es de 20 puntos en cada dimensión, por lo tanto, tenemos 400 puntos de validación.

En la siguiente tabla se muestra el error FVU para los datos de entrenamiento.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$4.363e^{-01}$	$1.727e^{-01}$	$2.945e^{-02}$	$5.035e^{-15}$	$5.112e^{-15}$	$5.114e^{-15}$
RF	$1.543e^{-01}$	$6.817e^{-15}$	$7.357e^{-15}$	$7.541e^{-15}$	$7.564e^{-15}$	$7.547e^{-15}$
SIF	$3.381e^{-15}$	$5.326e^{-15}$	$5.773e^{-15}$	$5.914e^{-15}$	$6.044e^{-15}$	$6.105e^{-15}$
CIF	$6.200e^{-01}$	$4.162e^{-01}$	$2.770e^{-01}$	$1.905e^{-01}$	$1.353e^{-01}$	$9.257e^{-02}$

Tabla 5.1: Datos de error de entrenamiento sin presencia de ruido seleccionando las mejores neuronas.

En la siguiente tabla se muestra el error FVU para los datos de validación:

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$4.615e^{-01}$	$1.822e^{-01}$	$3.489e^{-02}$	$2.645e^{-02}$	$2.645e^{-02}$	$2.645e^{-02}$
RF	$1.389e^{-01}$	$5.923e^{-15}$	$6.563e^{-15}$	$6.806e^{-15}$	$6.836e^{-15}$	$6.839e^{-15}$
SIF	$4.496e^{-15}$	$7.063e^{-15}$	$7.732e^{-15}$	$7.933e^{-15}$	$8.198e^{-15}$	$8.289e^{-15}$
CIF	$6.132e^{-01}$	$4.659e^{-01}$	$3.211e^{-01}$	$2.185e^{-01}$	$1.627e^{-01}$	$1.045e^{-01}$

Tabla 5.2: Datos de error de validación sin presencia de ruido para las mejores neuronas.

Gráficas:

En las siguientes cuatro figuras, se muestran la diferencia entre la función deseada y la salida de la red neuronal cuando tenemos 5 neuronas.

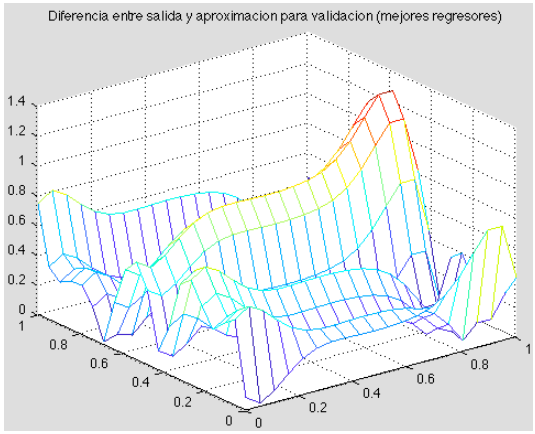


Figura 5.2: Diferencia para la función AF cuando tenemos 5 neuronas.

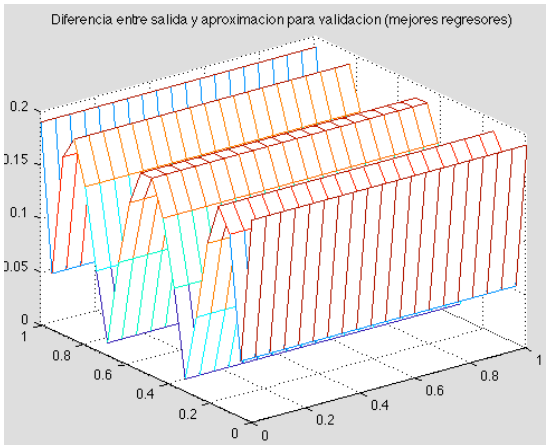


Figura 5.3: Diferencia para la función RF cuando tenemos 5 neuronas.

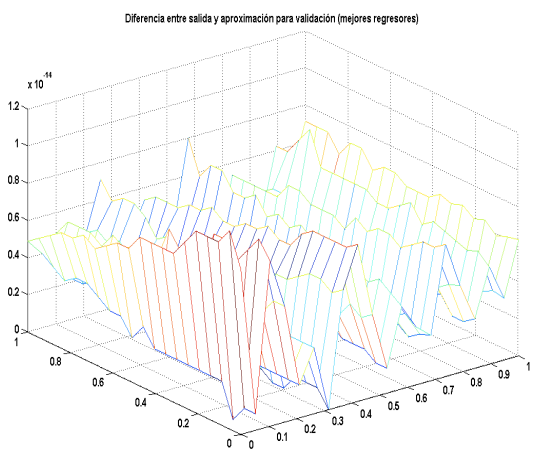


Figura 5.4: Diferencia para la función SIF cuando tenemos 5 neuronas.

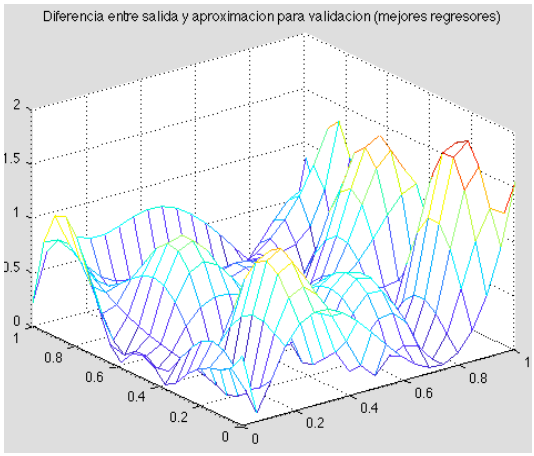


Figura 5.5: Diferencia para la función CIF cuando tenemos 5 neuronas

En las siguientes cuatro figuras, se muestran la diferencia entre la función deseada y la salida de la red neuronal cuando alcanzamos un error mínimo.

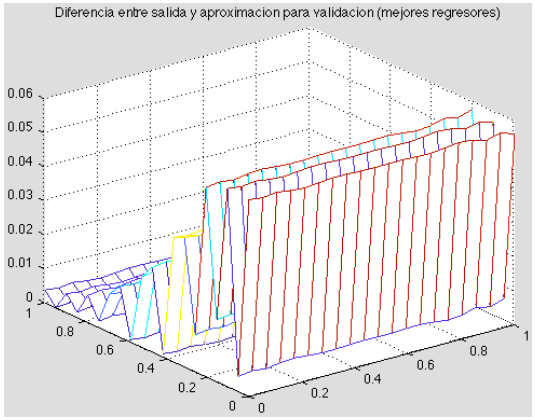


Figura 5.6: Función AF con 20 neuronas.

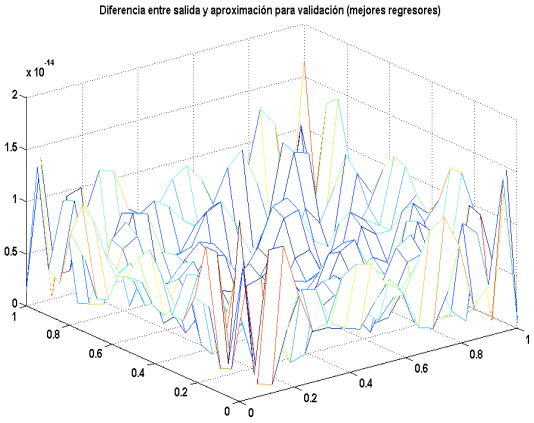


Figura 5.7: Función RF con 10 neuronas.

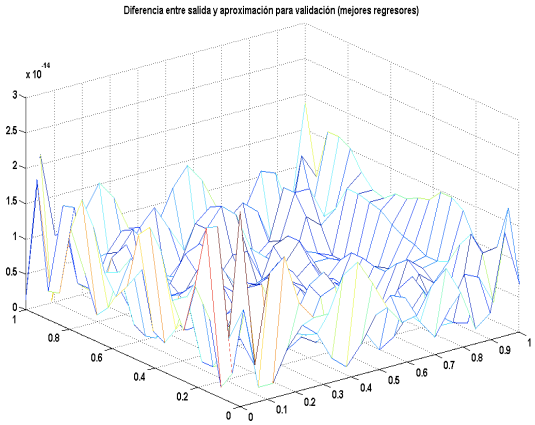


Figura 5.8: Función SIF con 10 neuronas.

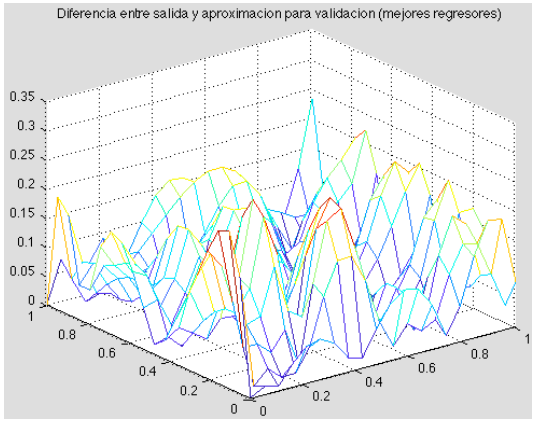


Figura 5.9: Función CIF con 30 neuronas

Podemos observar, que con 20 neuronas, en la función AF conseguimos un error de aproximación de validación mínimo (a partir de ese momento ya no cambia prácticamente).En cambio, en las funciones RF y CIF, el error de aproximación mínimo se consigue a partir de 10 neuronas. La función CIF es la mas difícil de aproximar ya que el error se reduce poco a poco, y no conseguimos errores tan reducidos como las otras tres funciones. El error mínimo se consigue con 30 neuronas. Sin embargo, podemos considerar que la aproximación es buena en todos los casos.

5.2. Estudio del comportamiento de la red neuronal con presencia de ruido en los datos de entrenamiento.

En este apartado, vamos a estudiar el comportamiento de la red neuronal con presencia de ruido, éste se le añade a los datos de entrenamiento, y el nivel del ruido es tal que la señal a ruido es de 6 dB. Para ello vamos a mostrar tablas de error FVU tanto en datos de entrenamiento como de validación, con pruebas de 5 hasta 30 neuronas, en intervalos de 5 en 5. El número de puntos de entrenamiento y validación, así como los intervalos, son los mismos que en el apartado anterior.

*Los datos de error FVU son en este caso, fruto del promedio de tres ejecuciones

.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$7.431e^{-01}$	$6.607e^{-01}$	$5.626e^{-01}$	$5.679e^{-01}$	$4.335e^{-01}$	$4.329e^{-01}$
RF	$7.537e^{-01}$	$6.786e^{-01}$	$5.713e^{-01}$	$5.138e^{-01}$	$4.624e^{-01}$	$4.628e^{-01}$
SIF	$7.429e^{-01}$	$7.042e^{-01}$	$5.549e^{-01}$	$4.909e^{-01}$	$4.383e^{-01}$	$3.567e^{-01}$
CIF	$8.539e^{-01}$	$6.444e^{-01}$	$6.462e^{-01}$	$5.375e^{-01}$	$4.615e^{-01}$	$4.489e^{-01}$

Tabla 5.3: Datos de error de entrenamiento con presencia de ruido.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$7.329e^{-01}$	$7.679e^{-01}$	$1.029e^{+00}$	$1.029e^{+00}$	$1.153e^{+00}$	$1.087e^{+00}$
RF	$6.741e^{-01}$	$8.075e^{-01}$	$1.136e^{+00}$	$1.196e^{+00}$	$1.177e^{+00}$	$1.262e^{+00}$
SIF	$9.147e^{-01}$	$1.004e^{+00}$	$1.478e^{+00}$	$1.716e^{+00}$	$1.658e^{+00}$	$1.859e^{+00}$
CIF	$8.541e^{-01}$	$1.136e^{+00}$	$1.211e^{+00}$	$1.208e^{+00}$	$1.397e^{+00}$	$1.426e^{+00}$

Tabla 5.4: Datos de error de validación con presencia de ruido.

Gráficas:

* Figuras con 5 neuronas estudiadas con la función aditiva, radial, de interacción simple e interacción complicada.

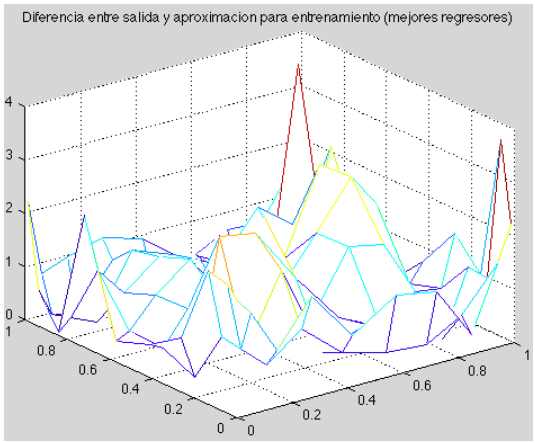


Figura 5.10: Función AF cuando tenemos 5 neuronas.

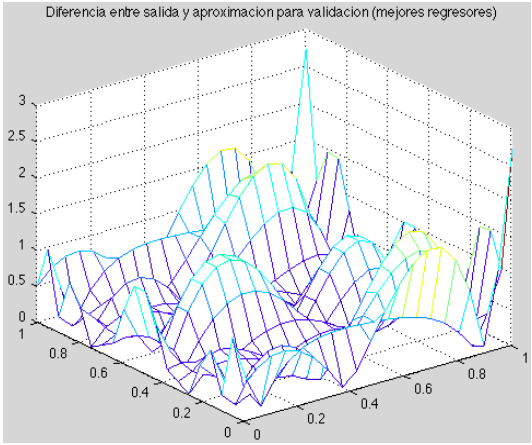


Figura 5.11: función RF cuando tenemos 5 neuronas.

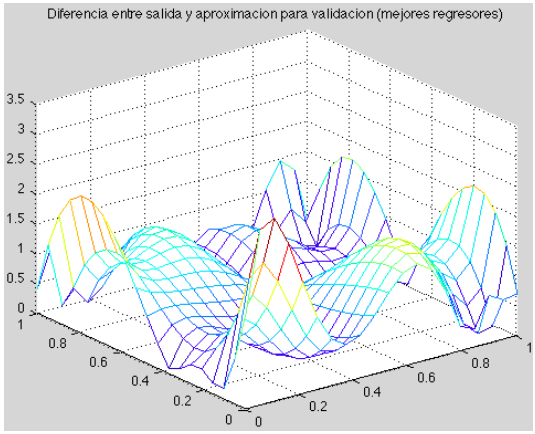


Figura 5.12: Función SIF cuando tenemos 5 neuronas.

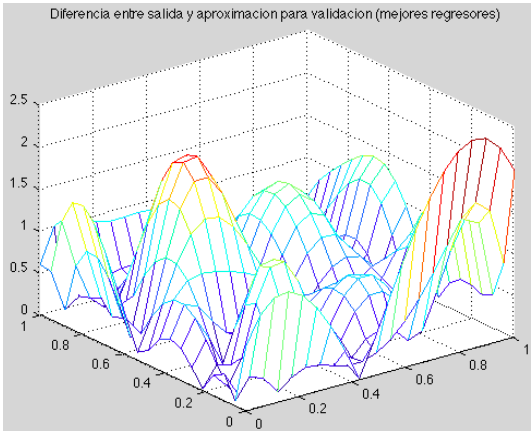


Figura 5.13: función CIF cuando tenemos 5 neuronas.

* Figuras con 15 neuronas estudiadas con la función aditiva, radial, de interacción simple e interacción complicada.

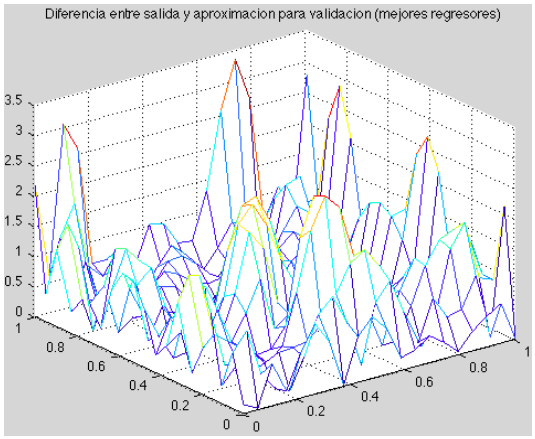


Figura 5.14: Función AF cuando tenemos 15 neuronas.

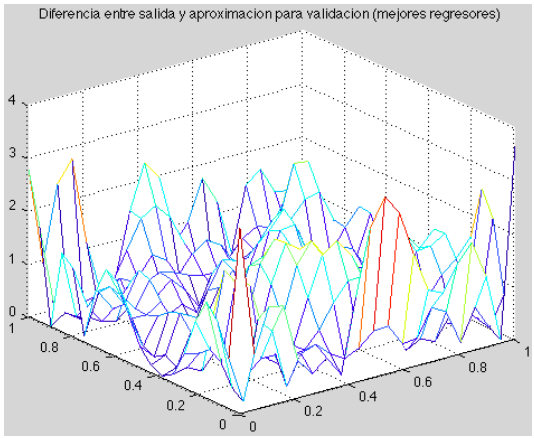


Figura 5.15: función RF cuando tenemos 15 neuronas.

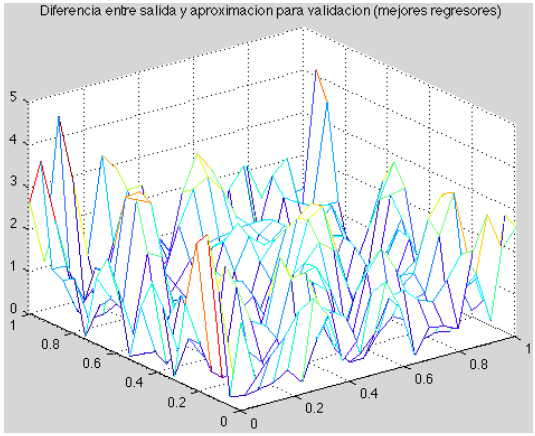


Figura 5.16: Función SIF cuando tenemos 15 neuronas.

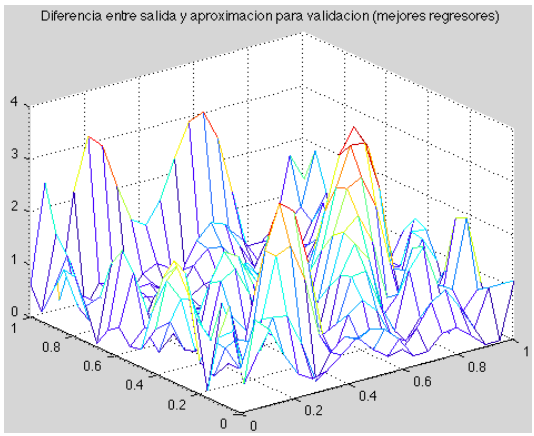


Figura 5.17: función CIF cuando tenemos 15 neuronas.

Para finalizar, podemos decir que, en todos los casos cuando aumentamos el número de neuronas también aumenta el error de aproximación, la red neuronal no está generalizando bien. Se observa que con 15 neuronas en lugar de mejorar el error de aproximación de validación, éste empeora. En cambio, si observamos la tabla del error de aproximación de entrenamiento, cuando el número de neuronas aumenta, el error de entrenamiento desciende. En el entrenamiento de estas redes neuronales lo que está ocurriendo es que hay una discrepancia significativa entre los datos de entrenamiento contaminados con ruido y los datos de validación sin ruido. La red aprende de unos datos que representan adecuadamente la señal deseada. Cuando se presenta un nuevo dato a la red, la salida que proporciona no es correcta ya que la red no sólo ha aprendido los datos correctos sino también el ruido que los acompañaba.

6. Redes neuronales de Chebychev multidimensionales en puntos arbitrarios.

Siguiendo las directrices del apartado 5.1, vamos a generar unas redes neuronales multidimensionales que en este caso serán capaces de operar en puntos arbitrarios. Para ello utilizaremos los polinomios de Chebychev unidimensionales de puntos arbitrarios desarrollados en el apartado 4.

6.1. Estudio del comportamiento de la red neuronal sin presencia de ruido en los datos de entrenamiento.

A continuación, vamos a estudiar el comportamiento de la red neuronal sin presencia de ruido. De forma análoga del capítulo anterior, vamos a mostrar tablas de error FVU tanto en datos de entrenamiento como de validación, con pruebas de 5 hasta 30 neuronas, en intervalos de 5 en 5.

Las funciones poseen un intervalo en el espacio de entrada $[0,1]^2$; el numero de puntos de entrenamiento es de 10 puntos en cada dimensión, luego tenemos 100 puntos en total. El numero de puntos de validación es de 20 puntos en cada dimensión, por lo tanto, tenemos 400 puntos de validación.

En la siguiente tabla se muestra el error FVU para los datos de entrenamiento.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$4.029e^{-01}$	$1.363e^{-01}$	$2.049e^{-02}$	$6.373e^{-16}$	$6.482e^{-16}$	$6.495e^{-16}$
RF	$1.492e^{-01}$	$6.329e^{-16}$	$5.637e^{-16}$	$5.647e^{-16}$	$5.849e^{-16}$	$5.835e^{-16}$
SIF	$5.329e^{-16}$	$6.107e^{-16}$	$6.314e^{-16}$	$6.334e^{-16}$	$6.292e^{-16}$	$6.271e^{-16}$
CIF	$6.366e^{-01}$	$4.376e^{-01}$	$3.017e^{-01}$	$2.080e^{-01}$	$1.481e^{-01}$	$9.886e^{-02}$

Tabla 6.1: Datos de error de entrenamiento sin presencia de ruido seleccionando las mejores neuronas.

En la siguiente tabla se muestra el error FVU para los datos de validación.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$4.008e^{-01}$	$2.294e^{-01}$	$1.524e^{-01}$	$1.539e^{-01}$	$1.539e^{-01}$	$1.539e^{-01}$
RF	$2.025e^{-01}$	$1.503e^{-01}$	$1.502e^{-01}$	$1.502e^{-01}$	$1.502e^{-01}$	$1.502e^{-01}$
SIF	$9.516e^{-16}$	$1.161e^{-15}$	$1.160e^{-15}$	$1.175e^{-15}$	$1.210e^{-15}$	$1.273e^{-15}$
CIF	$6.784e^{-01}$	$4.871e^{-01}$	$3.700e^{-01}$	$2.762e^{-01}$	$2.407e^{-01}$	$2.069e^{-01}$

Tabla 6.2: Datos de error de validación sin presencia de ruido para las mejores neuronas.

Gráficas:

Ahora, mostramos cualquiera de las gráficas de la matriz H de esta función. Ya que la matriz H debería ser igual para todas (todas poseen el mismo rango de trabajo [0,1] en ambas dimensiones y 10 puntos en cada dimensión).

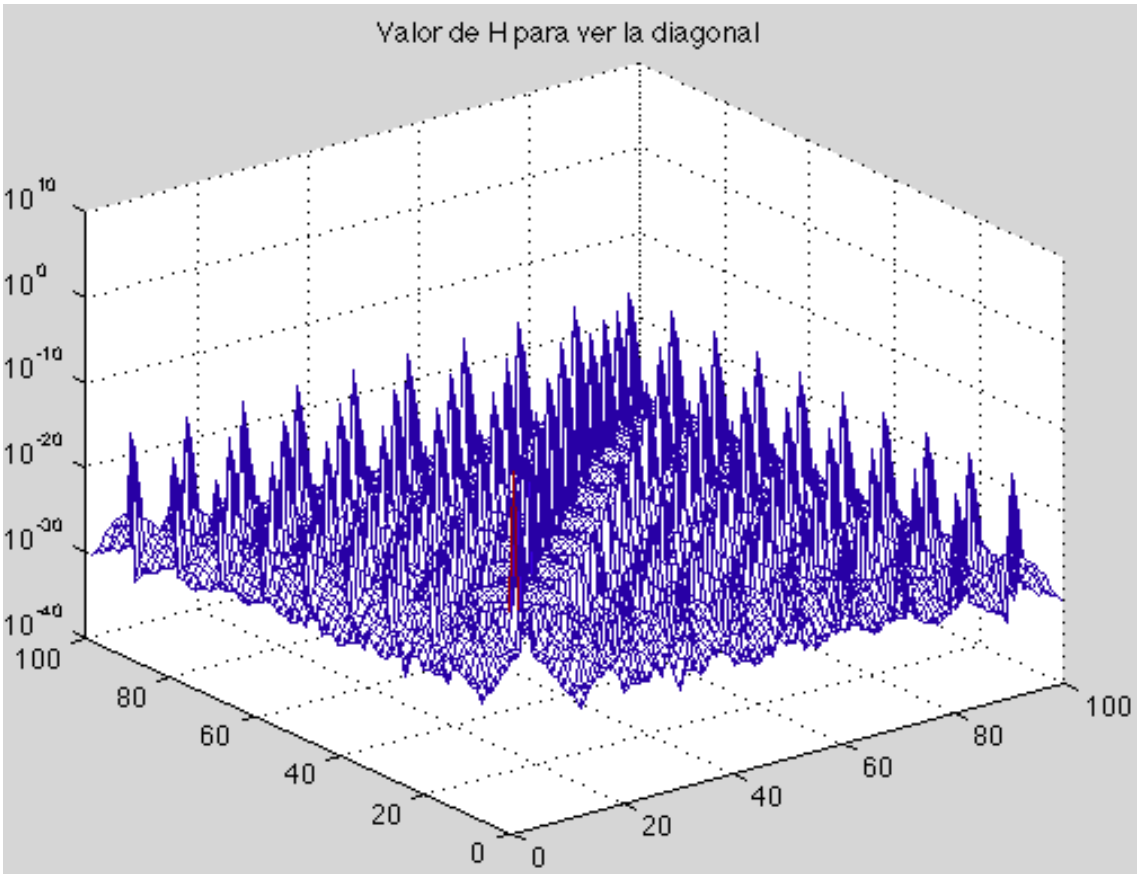


Figura 6.1: Matriz H en el intervalo [0,1] en ambas dimensiones, y 10 puntos en cada una de ella.

En las siguientes cuatro figuras, se muestra la diferencia entre la función deseada y la salida de la red neuronal cuando tenemos 5 neuronas.

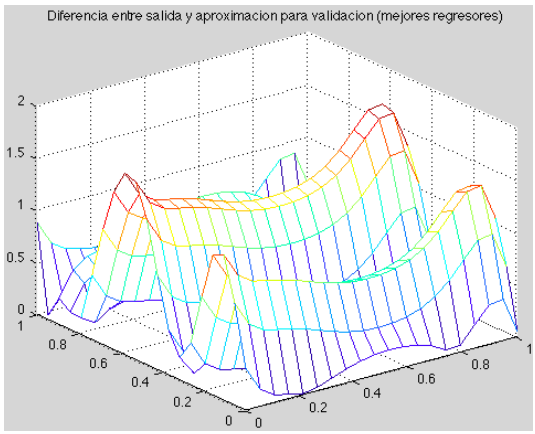


Figura 6.2: Diferencia para la función AF cuando tenemos 5 neuronas.

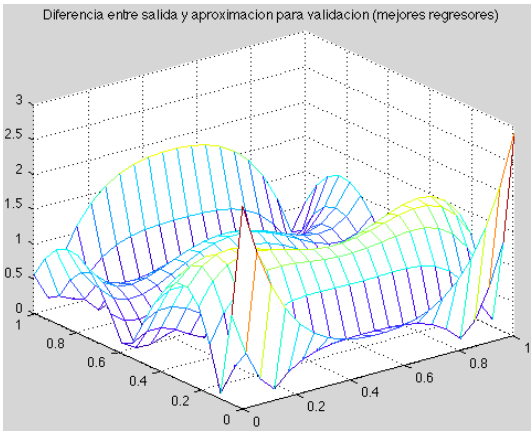


Figura 6.3: Diferencia para la función RF cuando tenemos 5 neuronas.

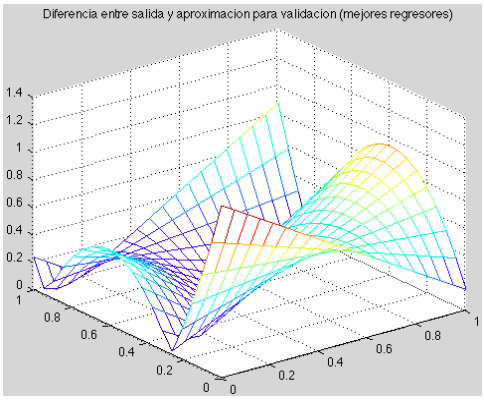


Figura 6.4: Diferencia para la función SIF cuando tenemos 5 neuronas.

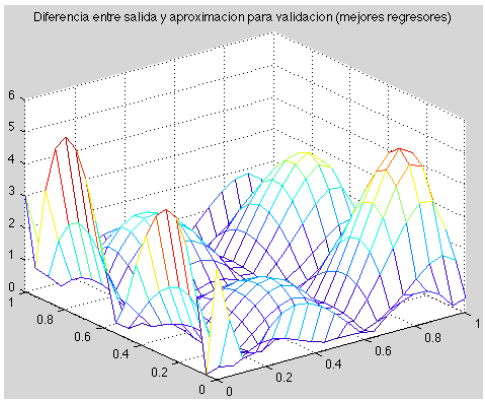
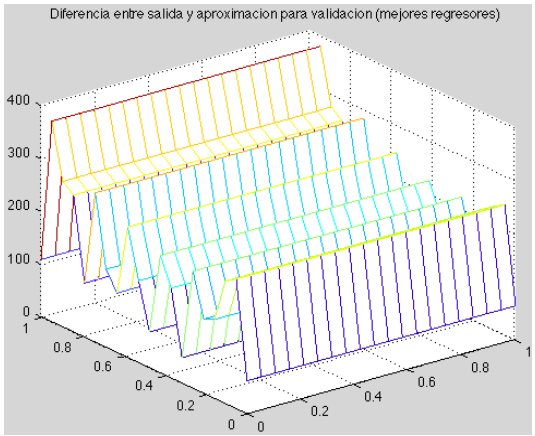


Figura 6.5: Diferencia para la función CIF cuando tenemos 5 neuronas

En las siguientes cuatro figuras, se muestran la diferencia entre la función deseada y la salida de la red neuronal cuando alcanzamos un error mínimo.



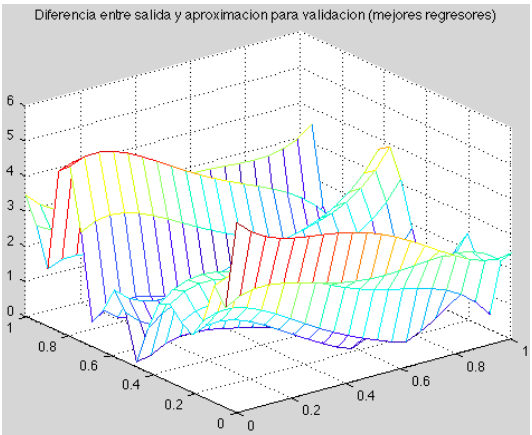


Figura 6.6: Función AF con 20 neuronas.

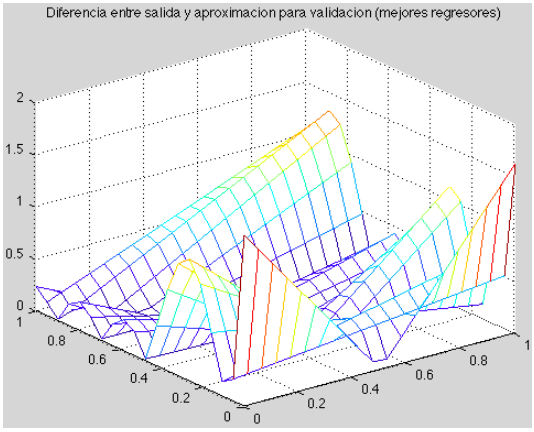


Figura 6.8: Función SIF con 10 neuronas.

Figura 6.67 Función RF con 10 neuronas

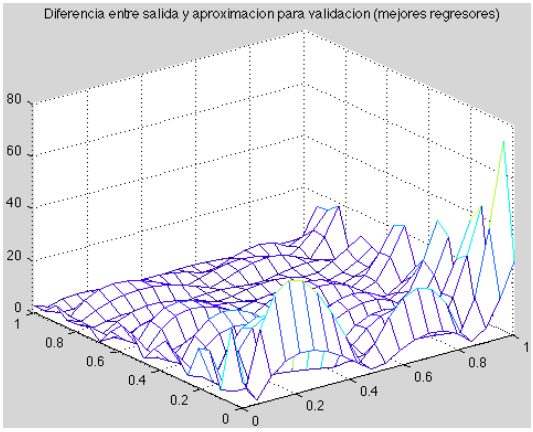


Figura 6.9: Función CIF con 30 neuronas

Podemos observar, que con 20 neuronas, en la función AF; y con 10 neuronas en las funciones RF y SIF, conseguimos un error de aproximación de validación mínimo (a partir de ese momento ya no cambia prácticamente). En cambio, con 30 neuronas la función CIF, el error se reduce poco a poco y no conseguimos errores tan reducidos como en las otras tres funciones.

Se puede apreciar lo mismo que en Chebychev no arbitrario. La función CIF es la más difícil de aproximar y en general en todas las funciones se alcanza un error de aproximación adecuado.

• Con datos de entrenamiento aleatorios, hemos realizado tres ejecuciones, y mostraremos la media de ellas:

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	1.607	0.791	0.298	$5.563e^{-07}$	$1.018e^{-06}$	$9.463e^{-07}$
RF	1.348	1.425	0.764	0.719	0.377	0.192
SIF	1.209	1.168	0.417	$1.265e^{-06}$	$2.619e^{-05}$	$3.19550e^{-05}$
CIF	3.557	3.517	2.399	1.089	0.848	0.817

Tabla 6.3: Datos de error de entrenamiento sin presencia de ruido en Chebychev arbitrario multidimensional.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	22.856	5.651	13.292	40.107	39.218	69.601
RF	1.838	15.930	2.566	6.809	10.397	4.611
SIF	6.178	52.338	20.685	81.038	50.329	66.915
CIF	44.909	287.103	416.179	37.352	9.729	223.384

Tabla 6.4: Datos de error de validación sin presencia de ruido en Chebychev arbitrario multidimensional.

En las siguientes cuatro figuras, se muestra la diferencia entre la función deseada y la salida de la red neuronal cuando tenemos 5 neuronas.

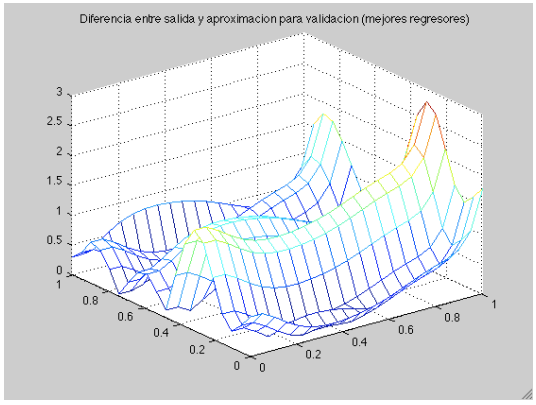


Figura 6.10: Diferencia para la función AF cuando tenemos 5 neuronas.

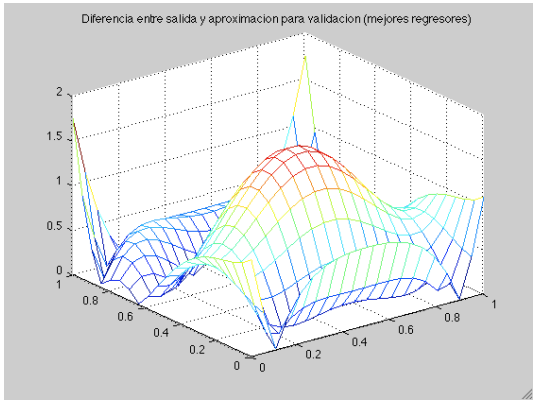


Figura 6.11: Diferencia para la función RF cuando tenemos 5 neuronas.

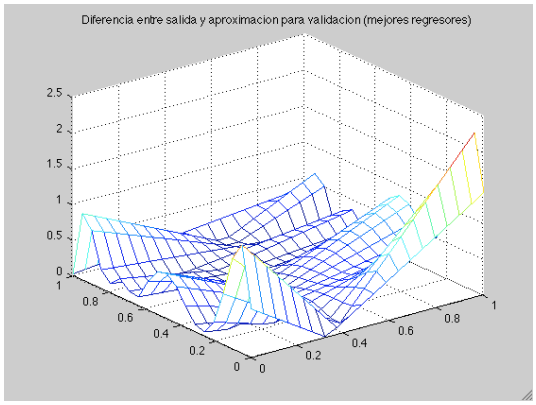


Figura 6.12: Diferencia para la función SIF cuando tenemos 5 neuronas.

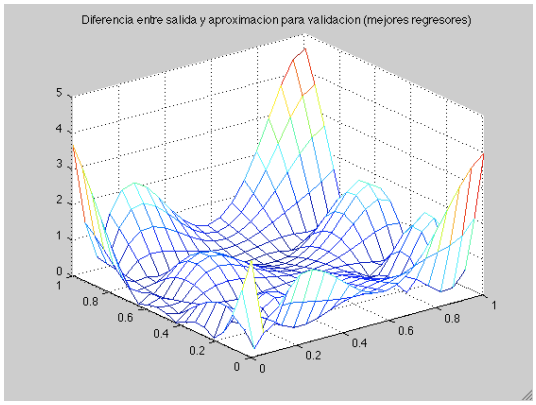


Figura 6.13: Diferencia para la función CIF cuando tenemos 5 neuronas

En las siguientes cuatro figuras, se muestran la diferencia entre la función deseada y la salida de la red neuronal cuando alcanzamos un error mínimo.

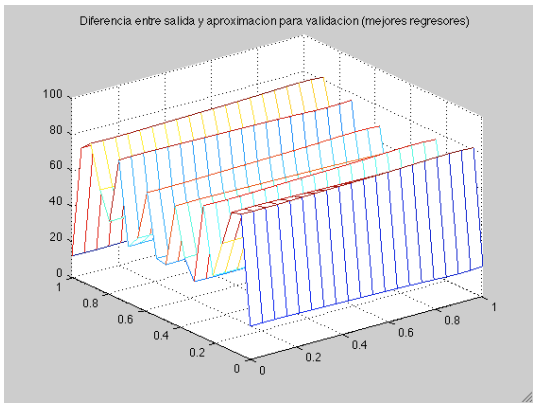


Figura 6.14: Función AF con 30 neuronas.

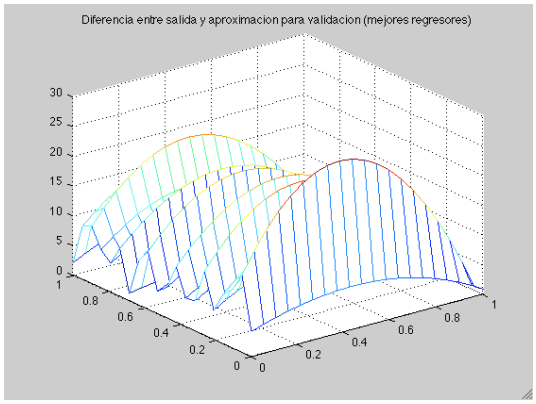


Figura 6.15: Función RF con 30 neuronas.

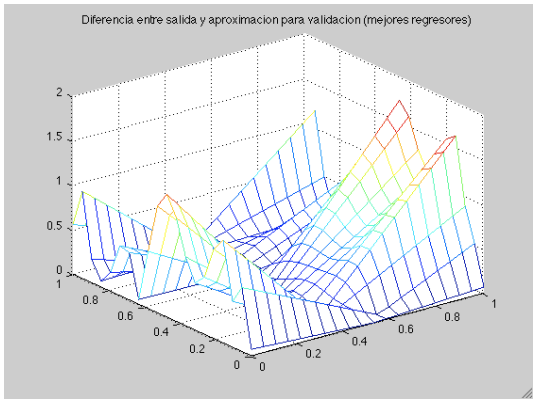


Figura 6.16: Función SIF con 10 neuronas.

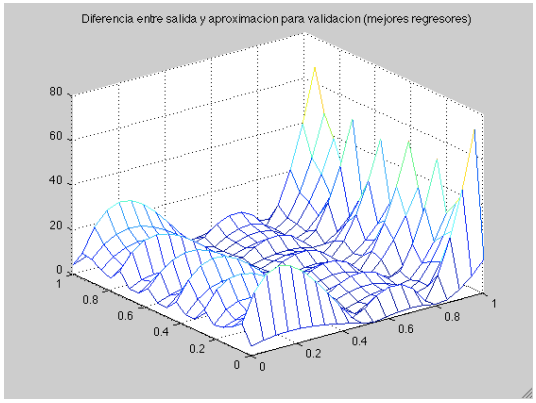


Figura 6.17: Función CIF con 30 neuronas.

Como vemos, el error de validación disminuye en menor medida que en el caso de tener puntos regulares. El aumento de neuronas no produce un gran descenso en los niveles de error de validación, experto en la función SIF, que como ya hemos visto es la mas fácil de aproximar. El hecho de contar con puntos de entrenamiento aleatorios provoca que haya zonas del espacio en las que no existen puntos de entrenamiento, y por tanto la red neuronal no aproxima correctamente en esas zonas.

6.2. Estudio del comportamiento de la red neuronal con presencia de ruido en los datos de entrenamiento.

En este apartado, vamos a estudiar el comportamiento de la red neuronal con presencia de ruido, éste se le añade a los datos de entrenamiento, y el nivel del ruido es tal que la señal a ruido es de 6 dB. Para ello vamos a mostrar tablas de error FVU tanto en datos de entrenamiento como de validación, con pruebas de 5 hasta 30 neuronas, en intervalos de 5 en 5. El número de puntos de entrenamiento y validación, así como los intervalos, son los mismos que en el apartado 5.

En la siguiente tabla se muestra el error FVU para los datos de entrenamiento en puntos regulares.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$7.702e^{-01}$	$6.673e^{-01}$	$6.488e^{-01}$	$5{,}356e^{-01}$	$3.674e^{-01}$	$4.281e^{-01}$
RF	$7.021e^{-01}$	$6.186e^{-01}$	$5.465e^{-01}$	$5.161e^{-01}$	$4.574e^{-01}$	$4.376e^{-01}$
SIF	$7.609e^{-01}$	$7.589e^{-01}$	$5.897e^{-01}$	$5.308e^{-01}$	$4.624e^{-01}$	$4.013e^{-01}$
CIF	$8.728e^{-01}$	$7.729e^{-01}$	$6.629e^{-01}$	$5.207e^{-01}$	$4.444e^{-01}$	$4.848e^{-01}$

Tabla 6.5: Datos de error de entrenamiento con presencia de ruido seleccionando las mejores neuronas en puntos regulares.

En la siguiente tabla se muestra el error FVU para los datos de validación en puntos regulares.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$7.546e^{-01}$	$6.379e^{-01}$	$2.037e^{-00}$	$4.782e^{-00}$	$2.503e^{-00}$	$2.172e^{-00}$
RF	$6.919e^{-01}$	$1.069e^{-00}$	$1.132e^{-00}$	$1.879e^{-00}$	$2.334e^{-00}$	$3.559e^{-00}$
SIF	$1.872e^{-00}$	$2.832e^{-00}$	$2.469e^{-00}$	$2.161e^{-00}$	$2.499e^{-00}$	$5.359e^{-00}$
CIF	$7.784e^{-01}$	$2.404e^{-00}$	$1.457e^{-01}$	$1.705e^{-00}$	$2.002e^{-00}$	$3.746e^{-00}$

Tabla 6.6: Datos de error de validación con presencia de ruido para las mejores neuronas en puntos regulares.

En las siguientes cuatro figuras, se muestra la diferencia entre la función deseada y la salida de la red neuronal cuando tenemos 5 neuronas.

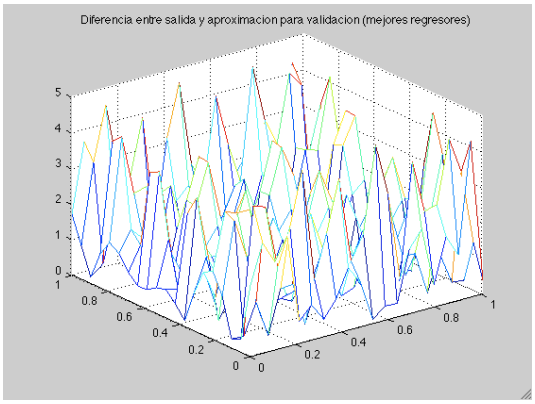


Figura 6.18: Diferencia para la función AF cuando tenemos 5 neuronas.

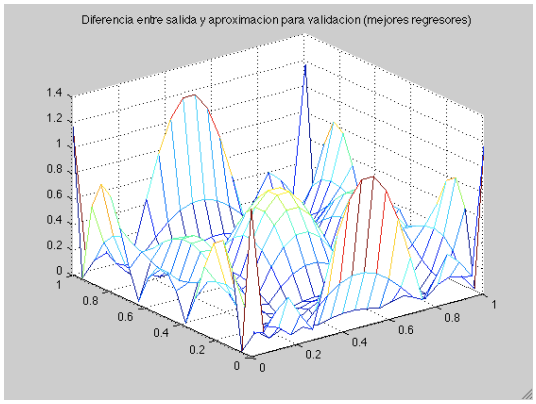


Figura 6.19: Diferencia para la función RF cuando tenemos 5 neuronas.

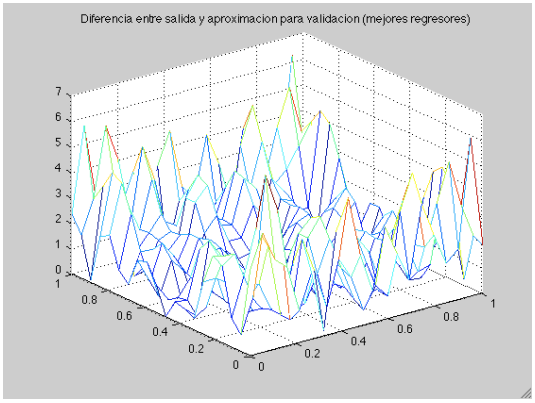


Figura 6.20: Diferencia para la función SIF cuando tenemos 5 neuronas.

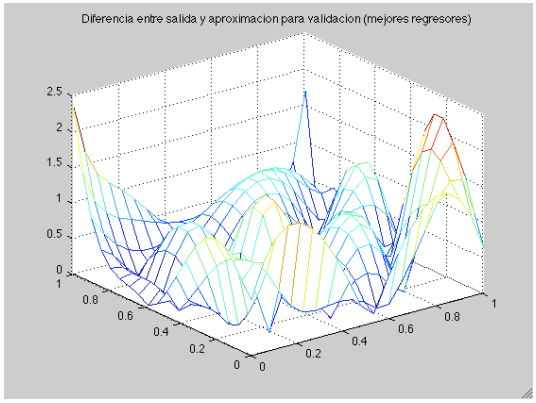


Figura 6.21: Diferencia para la función CIF cuando tenemos 5 neuronas

Para finalizar, podemos decir que, en todos los casos cuando aumentamos el número de neuronas también aumenta el error de aproximación, la red neuronal no está generalizando bien. Si observamos la tabla del error de aproximación de entrenamiento, cuando el número de neuronas aumenta, el error de entrenamiento desciende. En el entrenamiento de estas redes neuronales lo que está ocurriendo es que hay una discrepancia significativa entre los datos de entrenamiento contaminados con ruido y los datos de validación sin ruido. La red aprende de unos datos que representan adecuadamente la señal deseada. Cuando se presenta un nuevo dato a la red, la salida que proporciona no es correcta ya que la red no sólo ha aprendido los datos correctos sino también el ruido que los acompañaba.

• Con datos de entrenamiento aleatorios, hemos realizado tres ejecuciones, y mostraremos la media de ellas:

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	0.799	0.708	0.564	0.741	1.502	0.432
RF	0.839	0.703	1.259	0.521	82.989	0.415
SIF	0.925	0.732	0.648	0.878	0.887	0.462
CIF	0.837	1.075	0.923	0.571	1.133	0.621

Tabla 6.7: Datos de error de entrenamiento con presencia de ruido en Chebychev arbitrario multidimensional.

Función	5 neuronas	10 neuronas	15 neuronas	20 neuronas	25 neuronas	30 neuronas
AF	$0.587e03$	7.454	20.131	$1.874e06$	$2.209e09$	$0.773e03$
RF	26.414	6.273	$0.773e06$	$1.466e03$	$3.739e10$	$0.381e03$
SIF	$0.049e06$	$0.188e03$	$0.551e03$	$0.817e09$	$0.732e03$	$2.235e03$
CIF	1.981	$0.102e06$	$6.977e06$	33.290	$0.014e10$	$8.048e06$

Tabla 6.8: Datos de error de validación con presencia de ruido en Chebychev arbitrario multidimensional.

En las siguientes cuatro figuras, se muestra la diferencia entre la función deseada y la salida de la red neuronal cuando tenemos 5 neuronas.

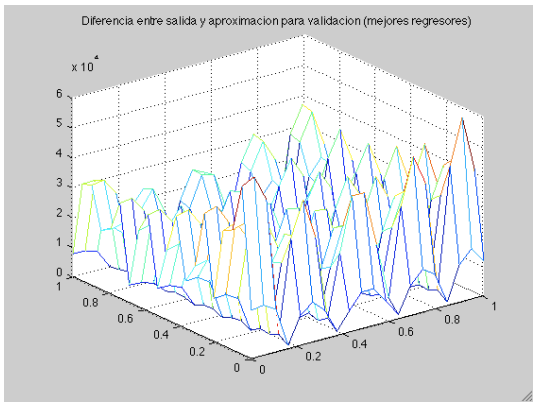


Figura 6.22: Diferencia para la función AF cuando tenemos 5 neuronas.

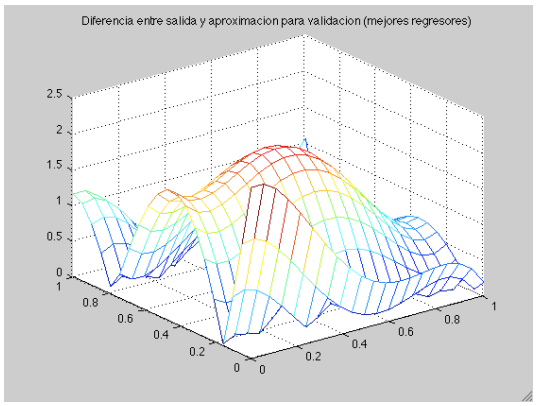


Figura 6.23: Diferencia para la función RF cuando tenemos 5 neuronas.

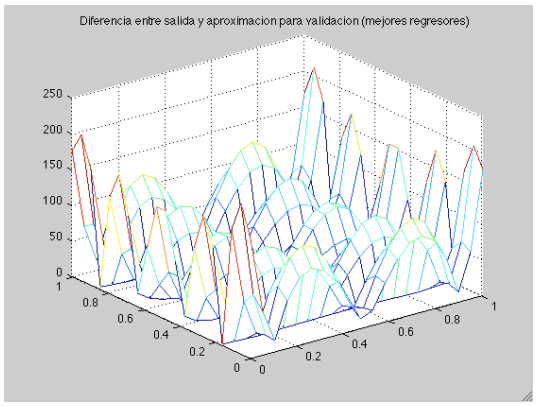


Figura 6.24: Diferencia para la función SIF cuando tenemos 5 neuronas.

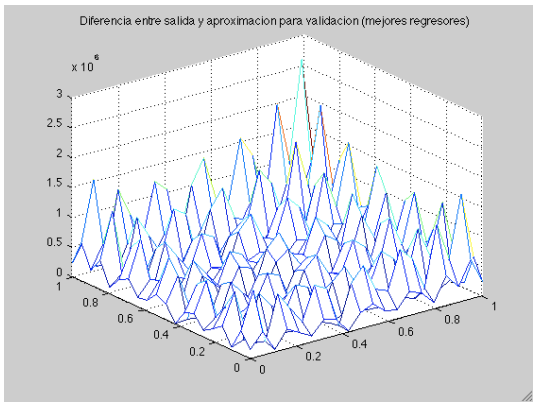


Figura 6.25: Diferencia para la función CIF cuando tenemos 5 neuronas

En este último caso, podemos observar de nuevo que, en todos los casos cuando aumentamos el número de neuronas también aumenta el error de aproximación, la red neuronal no está generalizando bien. Por ello solo hemos

mostrado las gráficas de error cuando tenemos cinco neuronas. Si observamos la tabla 6.8, se puede apreciar que el error de validación crece conforme aumenta el número de neuronas, por lo que podemos concluir de nuevo que la red neuronal no generaliza correctamente.

7. Conclusiones.

En este proyecto fin de carrera se ha estudiado el comportamiento de una red neuronal basada en polinomios de Chebychev. Se han empleado, tanto polinomios de Chebychev de primera especie como polinomios de Chebychev que son ortogonales en puntos arbitrarios. Esta red neuronal emplea el algoritmo de entrenamiento de mínimos cuadrados ortogonales (OLS); tal como se ha mostrado en este proyecto fin de carrera, este algoritmo tiene un funcionamiento óptimo si los polinomios o funciones neuronales utilizadas son ortogonales en los puntos de entrenamiento. Por este motivo se han empleado los mencionados polinomios de Chebychev. Los polinomios de Chebychev de primera especie son ortogonales en los llamados ceros de Chebychev mientras que el segundo conjunto de polinomios empleado son, como ya se ha mencionado, ortogonales en cualquier conjunto finito de puntos. El estudio realizado ha sido exhaustivo, ya que hemos estudiado la ortogonalidad de los polinomios, el número de puntos de entrenamiento necesarios, el numero de polinomios necesarios, así como el comportamiento de la red neuronal en presencia de ruido. Además, se han analizado tanto redes neuronales unidimensionales como multidimensionales.

En el análisis de los resultados obtenidos permite obtener diversas conclusiones. Se ha observado que los polinomios de Chebychev son siempre ortogonales sea cual sea el número de puntos de entrenamiento. Los polinomios de Chebychev son capaces de aproximar una gran cantidad de funciones; poseen una serie de características muy favorables para la aproximación de funciones. Por ello si podemos seleccionar los puntos de entrenamiento con libertad, la red neuronal de polinomios de Chebychev de primera especie, presenta un comportamiento muy eficiente. Posee gran capacidad de aproximación con un número mínimo de neuronas. Este hecho se ha comprobado tanto para redes neuronales unidimensionales como multidimensionales. La obtención de una red neuronal con un número mínimo de neuronas, es muy importante, ya que minimiza el tiempo de cálculo y facilita la implementación hardware de la neurona.

El único caso en el que esta red neuronal no aproxima correctamente la función deseada, es aquel en el que existe ruido en los puntos de entrenamiento. En esta situación, la red neuronal aproxima correctamente los puntos de entrenamiento, pero no aproxima de forma precisa los puntos de validación, es decir, no generaliza correctamente. El ruido altera la información contenida en los puntos de entrenamiento y por lo tanto la red neuronal no contiene información precisa sobre como es la función. Este comportamiento es común a otros tipos de redes neuronales que no son capaces de generalizar correctamente en presencia de ruido. La mayor limitación de las redes neuronales de Chebychev de primera especie radica en el hecho de que estos polinomios solamente son ortogonales en los ceros de Chebychev. Si no poseemos libertad para elegir los puntos de entrenamiento, esta red no se podrá utilizar de forma óptima.

La anterior limitación queda superada por los polinomios de Chebychev mostrados en el apartado 4. Teóricamente estos polinomios son ortogonales en cualquier conjunto de puntos, Sin embargo, al estudiarlos con detenimiento se ha observado que pierden la ortogonalidad cuando el número de entrenamiento

aumenta. A partir de 15 o 20 puntos de entrenamiento, dependiendo del intervalo de aproximación, los polinomios dejan de ser ortogonales. Si el número de puntos de entrenamiento no es grande, la red neuronal aproxima correctamente y tiene un tamaño mínimo; el algoritmo OLS selecciona los mejores polinomios de modo que se construye una red neuronal mínima. En el caso multidimensional se ha observado el mismo fenómeno, con 10 puntos de entrenamiento en cada dimensión, no se alcanzaba una ortogonalidad completa.

Los polinomios de Chebychev en puntos arbitrarios, aunque no alcanzan las características de aproximación de Chebychev de primera especie, poseen una buena capacidad de aproximación. Como se ha mostrado en la memoria, estas redes neuronales aproximan diversos tipos de funciones con gran precisión, tanto en el caso unidimensional como multidimensional. El error de aproximación es especialmente bajo cuando los puntos de entrenamiento son regulares, es decir, equidistantes. Cuando los puntos de entrenamiento son arbitrarios o aleatorios, existen situaciones en las que el error de aproximación es elevado; esto es debido a que existen zonas de la función sin apenas puntos de entrenamiento, lo que produce grandes errores de aproximación en esas zonas. Al igual que las redes neuronales de Chebychev de primera especie, estas redes no generalizan bien en presencia de ruido, incluso en el caso de emplear puntos regulares.

Como líneas futuras podemos mencionar el desarrollo de polinomios de Chebychev que sí sean ortogonales en conjuntos grandes de puntos de entrenamiento.

8. Bibliografía.

[1] Haykin, S. (1999). *Neural Networks, a comprehensive foundation*. Prentice Hall International.

[2] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[3] Rochester, N., Holland, J. H., Haibt, L. H., and Duda, W. L. (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on Information Theory*, IT-2:80–93.

[4] Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley.

[5] Uttley, A. M. (1956). A theory of the mechanism of learning based on the computation of conditional probabilities. *Proceedings of the First International Conference on Cybernetics*.

[6] Rosenblatt, F. (1956). The perceptron: a probabilistic model of information storage and organization in the brain. *Annals of Mathematical Statistics*, 27:832–837.

[7] Rosenblatt, F. (1960). On the convergence of reinforment procedures in simple perceptrons. *Cornell Aeronautical Laboratory Report*.

[8] Widrow, B.andHoff,M.E.(1960). Adaptives witching circuits. *IREWESCON coinvention record*, pages 96–104.

[9] Widrow,B.(1962). *Generalization and information storage in Networks of adaline'neurons' in Self-Organizing Systems*. Artech House.

[10] Minsky, M. L. and Papert, S. A. (1956). *Perceptrons*. Cambridge, MA, MIT Press.

[11] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79:2554–2558.

[12] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169.

[13] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning represetations of back-propagation errors. *Nature (London)*, 323:533–536.

[14] Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

[15] Vapnik, V.Ñ. (1992). Principles of risk minimization for learning theory. *Advances in Neural Information Processing Systems*, 4:831–838.

[16] Cortes, C. and Vapnik, V.Ñ. (1995). Support vector networks. *Machine Learning*, 20:273–297.

[17] Ma, L. and Khorasani, K. (2005). Constructive feedforward neural networks using hermite polynomial activation functions. *IEEE Transactions on Neural Networks*, 16(4):821– 833.

[18] Zhao, Y. and Atkeson, C. G. (1996). Implementing projection pursuit learning. *IEEE Transactions on Neural Networks*, 7(2):362–373.

[19] Hwang, J.-N., Lay, S.-R., Maechler, M., Douglas, R., and Schimert, J. (1994). Regression modeling in back-propagation and projection pursuit learning. *IEEE Transactions on Neural Networks*, 5(3):342–353.

[20] Tseng, C.-S. and Yang, S.-S. (1995). A new orthogonal neural network. *IEEE International Conference on Neural Networks*, 1(27):296–299.

[21] Yang, S.-S. and Tseng, C.-S. (1996). An orthogonal neural network for function approximation. *IEEE Transactions on systems, man, and cybernetics-part B: cybernetics*, 26(5):779–785.

[22] Namatame, A. and Ueda, N. (1992). Pattern classification with chebyshev neural networks. *International Journal of Neural Networks*, 3:23–31.

[23] Lee, T.-T. and Jeng, J.-T. (1998). The chebyshev-polynomials-based unified model neural networks for function approximation. *IEEE Transactions on systems, man, and cybernetics-part B: cybernetics*, 28(6):925–935.

[24] Weng, W.-D., Yang, C.-S., and Lin, R.-C. (2007). A channel equalizer using reduced decision feedback chebyshev functional link artificial neural networks. *Information Sciences*, 177(13):2642–2654.

[25] Purwar, S., Kar, I.Ñ., and Jha, A.Ñ. (2007). On-line system identification of complex systems using chebyshev neural networks. *Applied soft computing*, 7(1):364–372.

[26] Chen,C.-S.andTseng,C.-S.(2004).Performance comparison between the training method and the numerical method of the orthogonal neural network in function approximation. *International journal of intelligent systems*, 19(12):1257–1275.

[27] Cauwenberghs, G. (1993). A fast stochastic error-descent algorithm for supervised learning and optimization. In Hanson, S.J., Cowan, J.D., and Giles, C.L., editors, Advances in Neural Information Processing Systems 5, pages 244 – 251, San Mateo, California. Morgan Kaufmann Publishers.

[28] Chen, S., Cowan, C. F.Ñ., and Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302– 309.

[29] Sherstinsky, A. and Picard, R. W. (1996). On the efficiency of the orthogonal least squares training method for radial basis fucntion networks. *IEEE Transactions on Neural Networks*, 7(1):195–200.

[30] Press,W.H.,Teukolsky,S.A.,Vetterling,W.T.,andFlannery,B.P.(1997).*Numerical recipes in Fortran 77, Second edition*. Cambridge University Press.

[31] Phillips, G. M. (2003). *Interpolation and approximation by polynomials*. Springer-Verlag, New York.

[32] "The Use of Chebyshev Polynomials Orthogonal on a Finite a Arbitrary System of Points for Interpolating Changes in Nematic Isotropic Liquid Phase Transition Temperatures in Hololog ous Series "Russian Journal of Physical Chemistry 2006 – Vol.80, pp 122-127],Plaides Publisting.